

SOURCE-AND-SINK METHOD OF SOLUTION OF
MOVING BOUNDARY PROBLEMS

BY

MEHDI AKBARI

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1993

TO MY WIFE VAHIDEH, AND DAUGHTERS SARA AND MONA

ACKNOWLEDGEMENTS

I am most grateful to the chairman of my supervisory committee, Dr. C. K. Hsieh, who provided the inspiration for this study as well as many hours of guidance and encouragement. His enthusiasm has been contagious, and I hope to emulate his attitude in the future.

I wish to express my sincere thanks to Drs. R. A. Gater, G. G. Emch, R. Abbaschian, and H. A. Ingley whose unselfish good counsel and understanding have been essential in this study.

Last but by no means least, I would like to thank my wife, Vahideh, for her love and spiritual support during this period of time of work on the dissertation.

TABLE OF CONTENTS

	<u>Page</u>
ACKNOWLEDGEMENT	iii
LIST OF TABLES	vi
LIST OF FIGURES	vii
NOMENCLATURE	ix
ABSTRACT	xi
 CHAPTERS	
I INTRODUCTION	1
II LITERATURE REVIEW	8
III SOLUTION OF INVERSE STEFAN PROBLEMS BY A SOURCE-AND-SINK METHOD	21
3.1 Motivation	21
3.2 General Analysis	25
Pre-melt Stage	28
Melting Stage	28
3.3 Example Problems	31
3.4 Numerical Solution	33
3.5 Critique of The Method	39
3.6 Results and Discussion	40
3.7 Extension and Concluding Remarks	56
IV SOLUTION OF ABLATION PROBLEMS WITH ONE MOVING BOUNDARY BY A SOURCE-AND-SINK METHOD	58
4.1 Solution Methodology	58
Pre-ablation Stage	59
Ablation Stage	62
Equivalent Problem	63
4.2 Illustrative Examples	65
4.3 Numerical Solution of Ablated Front	67
4.4 Results and Discussion	69
V SOLUTION OF ABLATION PROBLEMS WITH TWO MOVING BOUNDARIES BY A SOURCE-AND-SINK METHOD	90

	<u>Page</u>
5.1 General Analysis	91
Pre-melt Stage	91
Melting Stage	94
Ablation Stage	95
Equivalent Problem	98
5.2 Examples	101
5.3 Numerical Solution of The Ablation Problem	104
5.4 Numerical Solution of Temperature and Energy Storage	109
5.5 Numerical Examples	110
VI DISCUSSION AND RESULTS	130
APPENDICES	
A EXTENSION OF THE SSM	135
B SSM FORTRAN PROGRAM FOR INVERSE STEFAN PROBLEM	136
C SSM FORTRAN PROGRAM FOR INVERSE STEFAN PROBLEM	147
D SSM FORTRAN PROGRAM FOR INVERSE STEFAN PROBLEM	154
E SSM FORTRAN PROGRAM FOR INVERSE STEFAN PROBLEM	165
F SSM FORTRAN PROGRAM FOR ABLATION PROBLEM	172
G SSM FORTRAN PROGRAM FOR COMBINATION PROBLEM	186
REFERENCES	223
BIOGRAPHICAL SKETCH	229

LIST OF TABLES

<u>Table</u>	<u>Page</u>
3.1 Expressions to account for effects of boundary conditions	23
3.2 Properties of aluminum	41
3.3 Conditions tested in eight examples	42
3.4 Comparison between true and retrieved conditions for first Stefan-Neumann example solved for boundary temperature	43
3.5 Comparison between true and retrieved conditions for second Stefan-Neumann example solved for boundary flux then temperature	46
3.6 Comparison between true and retrieved conditions for third Stefan-Neumann example solved for boundary temperature	48
3.7 Comparison between true and retrieved conditions for fourth Stefan-Neumann example solved for boundary flux then temperature	49
3.8 Comparison between true and retrieved conditions for fifth inverse example problem solved for temperature	50
3.9 Comparison between true and retrieved conditions for sixth inverse example problem solved for heat flux	51
3.10 Comparison between true and retrieved conditions for seventh inverse example problem solved for temperature	52
3.11 Comparison between true and retrieved conditions for eight inverse example problem solved for heat flux	53
4.1 Conditions tested in four examples	70
5.1 Conditions tested in three examples	111

LIST OF FIGURES

<u>Figure</u>		<u>Page</u>
3.1	System analyzed	27
3.2	Linearization of solid-liquid interface position curves for the numerical solution	35
4.1	System analyzed	61
4.2	Trend of ablated surface position for a medium initially at phase-change temperature imposed with a constant heat-flux condition	73
4.3	Accuracy, stability and convergency test of the SSM in the solution of ablation for a medium initially at phase-change temperature imposed with a constant heat-flux condition	75
4.4	Stability and convergency test of the SSM in the solution of ablation for a subcooled medium imposed with a constant heat-flux condition	78
4.5	Comparison of ablated surface position with two methods in the literature for a subcooled medium imposed with a constant heat-flux condition	80
4.6	Comparison of ablated surface position with the moment method in the literature for a subcooled medium imposed with a linear heat -flux condition	83
4.7	Stability and convergency test of the SSM in the solution of ablation for a subcooled medium imposed with a linear heat-flux condition	85
4.8	Comparison of ablated surface position with the moment method in the literature for a subcooled medium imposed with a quadratic heat-flux condition	87
4.9	Stability and convergency test of the SSM in the solution of ablation for a subcooled medium imposed with a quadratic heat-flux condition	89

<u>Figure</u>	<u>Page</u>
5.1	System analyzed93
5.2	Linearization of solid-liquid interface and ablated surface position curves for the numerical solution106
5.3	Trends of solid-liquid interface and ablated surface positions for a combination problem of subcooled medium imposed with a constant heat-flux condition114
5.4	Trends of solid-liquid interface and ablated surface positions for a combination problem of subcooled medium imposed with a linear heat-flux condition116
5.5	Trends of solid-liquid interface and ablated surface positions for a combination problem of subcooled medium imposed with a quadratic heat-flux condition118
5.6	Temperature distributions in the medium at different times during ablation for a combination problem of subcooled medium imposed with a constant heat-flux condition120
5.7	Stability and convergency test of the SSM in the solution of ablation for a combination problem of subcooled medium imposed with a constant heat-flux condition122
5.8	Stability and convergency test of the SSM in the solution of ablation for a combination problem of subcooled medium imposed with a linear heat-flux condition124
5.9	Stability and convergency test of the SSM in the solution of ablation for a combination problem of subcooled medium imposed with a quadratic heat-flux condition126
5.10	Overall accuracy test of the SSM in the solution of the combination problem of subcooled medium imposed with a constant heat-flux condition129

NOMENCLATURE

c	Specific heat
E	Equation
F	Temperature function used for boundary condition
G	Flux function used for boundary condition, Green's function, elements in coefficient matrix
g	Heat flux
k	Thermal conductivity
L_f	Latent heat of fusion
L_v	Latent heat of vaporization
\hat{n}	Normal unit vector
r	Distance from sink or source to sense point
R_1, R_2	Interface positions
Ste	Stefan number
s	Dummy variable for integration, interface speed
T	Temperature, temperature vector
t	Time
v	velocity of the interface
x	Position, source point
α	Thermal diffusivity
δ	Dirac delta function, Kronecker delta

ρ Density
 τ Dummy variable for time

Subscripts and Superscripts

m,v Phase change
 N_1, N_2 Upper time limit index
n Time index
t Transition
1,2 Melt and vapor fronts
0 Initial time

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

SOURCE-AND-SINK METHOD OF SOLUTION OF
MOVING BOUNDARY PROBLEMS

By

Mehdi Akbari

May 1993

Chairman: Dr. Chung K. Hsieh
Major Department: Mechanical Engineering

Source-and-sink methods have been developed for the solution of inverse diffusion problems, ablation problems, and combination of ablation and Stefan problems. Green's functions have been used and integrodifferential equations are derived for the interface positions and the temperature in the phase-change medium. These equations are solved by using local linearization of the interface position and the boundary heat flux if it were treated as unknown. The results have shown to be accurate, convergent, and stable.

The methods developed for the solution of the inverse diffusion problems have been used to find the boundary conditions for the inverse Stefan problems. They are solved by two approaches: a series solution approach and a time incremental approach. Both have shown to be useful to find the boundary conditions without reliance on the flux information to be supplied at both sides of the phase change interface. The methods are efficient in that they require fewer equations to be solved for the

unknowns. The algorithms can also be easily developed for the solution of the problems.

In the use of the source-and-sink method to analyze the ablation and combination problems, heat transfer is solved in a fixed domain so that the original heat-flux condition that is imposed on the moving boundary is taken to be the condition imposed on the solid-vapor or liquid-vapor interface. This flux condition is then used together with the temperature at the interface to solve for the interface position as well as the condition imposed on the fixed boundary. Finally, these positions and conditions are used in the temperature equation to complete the solution. The method has been used successfully in solving examples encompassing one-, two-, and three-phase ablation with the boundary of the medium imposed with constant, linear, and quadratic flux conditions. Numerical solutions for these examples as well as the trends of the moving boundary positions are discussed in detail.

CHAPTER I INTRODUCTION

Phase changes occurs when a body is exposed to a large heat flux so that it melts or vaporizes at the surface. With continuous heating, the melting or vaporization front moves inward with time. Since the position of the phase-change interface is unknown a priori, this type of problem is commonly known as the moving boundary problem in the heat transfer literature. Phase change problems find applications in thermal energy storage, material treatment and processing, thermal protection of re-entry vehicles in space technology, laser drilling and cutting in manufacturing, freezing and thawing in soils and foodstuff, and photocoagulation in opthalmological procedures, among others.

It is difficult to solve a phase-change problem because of the presence of the nonlinear condition at the interface. Usually for such a problem, the number of phases encountered in the medium depends on the ambient pressure and the temperature range that appears in the medium. Thus for a medium heated or cooled below the triple point of the substance, only sublimation and fusion may take place. On the other hand, for phase change above the triple point, solid, liquid, and gas states may appear simultaneously, and this occurs for a medium of low thermal conductivity and exposed to a large heat flux. In practice, the changed phase may be removed as

soon as it is formed. This occurs in the re-entry where the melted substance is removed by a fluid dynamic force. This material removal may also be effected naturally such as in sublimation, where the sublimed vapor is dissipated by diffusion to the surroundings. In either event, the imposed heat flux follows the boundary motion, and such problems are commonly referred to as ablation problems in the literature.

A different state of affairs is encountered when the changed phase remains stationary and continues to occupy the space taken by its previous state. Thus for example in melting a solid, if the melt is not removed by an external force, it will adhere to the solid from which it changes phase. Then in the absence of convection in the melt, the heat transfer in the medium can be analyzed by solving a Stefan problem in recognition of Stefan who first presented a formal solution of the problem in the open literature (see next chapter for a survey of the literature). The Stefan problem is basically different from the ablation problem in that, in the former, the problem is solved in a fixed domain, and the boundary condition is always imposed on the fixed boundary whereas, in the latter, the imposed condition follows the boundary motion. Thus in the ablation problem, the domain for analysis diminishes in size with time, and the imposed condition is acting on the receding boundary due to the shrinkage of the medium. There have been numerous efforts in the literature for the solution of the Stefan problems. Much less research, however, has been devoted to the solution of the ablation problems. Also for those ablation problems that have been solved and documented in the literature,

most are for the analysis of generic ablation in which only one moving boundary appears in the medium. There has been a lack of study for problems involving two moving boundaries such as those found at the solid-liquid and liquid-vapor interfaces. They are encountered when the solid, liquid, and gas states appear simultaneously in the medium.

Three phase-change problems will be solved in this dissertation. In the first problem, an inverse Stefan problem will be solved. This problem differs from a conventional Stefan problem in the sense that, for the conventional problem, the condition imposed on the boundary is fully specified. This condition is then used together with others to solve for interface motion as well as the temperature distribution in the medium. For the inverse problem, however, the interface motion is given, and this motion is used as a part of the input to solve for the boundary condition that is necessary to provide for this interface motion. This problem is found in material treatment and processing, where the property of the material may be a function of the phase-change rate at the interface. Solution of the problem thus enables a better control of the property of the material.

It should be noted that the inverse Stefan problem described above has recently been solved by Zabaras et al. [1] in the literature. However, the method developed for solution in this dissertation is more efficient because it requires no information for the heat flux at the interface. As will be shown, two methods are developed for the solution of the problem: one expands the condition in a polynomial or an infinite series, and the other uses

a time incremental method. The former utilizes the interface position information at discrete times to solve for the coefficients in the polynomial or series expansion, while the latter uses the interface position at consecutive times to solve for the conditions incrementally. Both can be used to solve for the conditions accurately as supported by eight examples encompassing the search for constant and time-variant temperature and flux conditions at the boundary. The methods are also well adapted to a numeral solution.

The second phase-change problem solved in this work is an ablation problem that contains only one moving boundary. This problem is solved by a new method, which treats the problem in a fixed domain. In this method, the ablated phase is considered to remain in the space that is ablated in the process. As such, the moving boundary can be taken to be an interior phase-change interface, and the heat flux that is derived at this interface can be used to match the condition imposed at the moving boundary. Then with the additional prescription of the temperature for phase change at this interface, the flux condition there can be used to determine the position of the interface as well as the hypothetical condition that is imposed at the fixed boundary. Finally these positions and conditions can be used in the temperature equation to complete the solution.

Solution of the ablation problems has rarely been attempted in a fixed domain. The method presented in this work thus represents a new endeavor, which will be shown to be simpler than most methods described in the literature. To validate the method used in the solution of the ablation problems, four examples are provided and

they include the analysis of one- and two-phase ablation imposed with constant and time-variant flux conditions at the moving boundary. Of those four, the ones imposed with constant heat flux can be checked with an exact solution. A one-phase ablation problem is thus selected for checking the solution over the entire period of the ablation, while a two-phase ablation problem is selected for testing the solution in a quasi-steady state. Both yield accurate results as shown later in this dissertation. Convergence and stability tests have also been made rendering further assurance for the success of the method.

The good results obtained in the second problem described above provide impetus for the solution of the third problem that addresses ablation with two moving boundaries. This is a combination of the ablation and Stefan problems, an area that has rarely been studied in the literature yet is important in practice when solid, liquid, and gas states appear simultaneously in the medium. Again the problem is solved in a fixed domain, and the moving boundary is treated as an interior phase-change interface. The flux condition at this boundary is again used with the temperatures at the interfaces to determine the heat flux at the fixed boundary as well as the interface positions to complete the solution. Unlike its predecessor for which two simultaneous integrodifferential equations are solved simultaneously for each time step, for the combination problem, three equations will be solved simultaneously. Again the results are good as evidenced by three examples including imposition of constant, linear, and quadratic flux conditions.

A source-and-sink method will be used for the solution of all the problems in this work. This method has recently been subjected to a through development [2-5], where the method has shown to be particularly suited for the solution of the phase-change problems addressed in this work. In this method, the problem is solved by using one set of governing equation, initial, and boundary conditions, whereas in the conventional methods developed in the literature, the problem is solved by using different sets of equations, with each set focused on the solution of one phase region. The conventional methods thus cannot compete with the source-and-sink method for simplicity. Moreover, in the source-and-sink method, only one temperature equation will be derived; whether it is in the solid or liquid region depends on the position that is assigned in the equation. This is in sharp contrast to the conventional methods, in which several temperature equations must be derived, again one for each phase region. The time and effort saved in the source-and-sink method can thus be readily appreciated.

With the inclusion of the three separate problems in this dissertation, the presentation of materials follows a nontraditional approach. Thus one chapter will be devoted for the presentation of each problem; and the chapters will be self-complete with the coverage of the statement of the problem, general solution methodology, analysis, examples, and results and discussion. It is felt that, only through such an organization, the material can be presented in a coherent manner without severe fragmentation. In what follows in the next chapter, a literature review will be given first. It is then followed by the presentation of the three

problems. Finally, conclusions and recommendations will be given to conclude this dissertation.

As usual, all computer programs developed in this work are compiled in the appendix. It is hoped that the insight gained from this research will be useful in the analysis of the phase change in the future.

CHAPTER II LITERATURE REVIEW

Heat conduction with phase change constitutes a large class of moving boundary problems. These problems are commonly referred to as Stefan problems and were first solved in 1831 by Lamé and Clapeyron [6], who determined the thickness of the solid crust by freezing of water under a constant temperature condition. They were able to find the thickness to be proportional to the square root of time but did not find the constant of proportionality. Some thirty years later, Neumann presented an exact solution to the problem in his unpublished lecture notes, and for the first time in history, the Stefan problem was solved in its entirety [7-8]. Neumann's solution was for the phase change in a semi-infinite space imposed with a constant temperature condition. His work is important because, irrespective of the numerous efforts made by many others in search for additional exact solutions, Stefan problems that can be solved exactly today remain those that were originally analyzed by Neumann some one hundred and thirty years ago.

While the exact solution to the Stefan problem was first obtained by Neumann, the problem has been named after Stefan in recognition of his published work appearing in the open literature in 1889 [6-9]. Practically the same problem was solved by Stefan, who expressed the temperature of the phase-change medium in terms of

a similarity variable, x/\sqrt{t} , and the position of the moving boundary to be proportional to the square root of time. Thus, in essence, the moving boundary was solved by means of a similarity transformation. Since the transformation can not be used for general problems, it has now been firmly established that, for a Stefan problem that can be solved exactly, it must be in an unbounded domain, of medium of constant properties, and imposed with a constant temperature condition.

With the severe limitation imposed by the similarity transformation, it is not surprising to see a wide variety of approximate solutions developed in the literature. Books and monographs have been prepared for presentation of these methods. In what follows, some popular solution methods will be briefly reviewed; interested readers are referred to 10-14 for details.

One of the early methods developed for the solution of the Stefan problems imposed with heat flux and convection conditions is the power-series expansion method. In this method, the interface position and the temperature in the medium are expanded in terms of power series [10] or complimentary error functions [11-14]. These series are then substituted into the governing equation and boundary conditions to determine the coefficients in the series expansion.

Power series method works well for the short-time solutions. At that time, the series converge rapidly, and only a few terms of the series are sufficient to yield accurate results. The method also works for the solution in the neighborhood of singularities that may occur as a phase degenerates. For large time, more terms are needed in the series, and the method rapidly loses its appeal.

Stefan problems can also be solved by reducing them to integrodifferential equations. Lightfoot [15] took the solidification as a moving heat source front. With the use of Green's function, he was able to derive two integrodifferential equations, one for the temperature and the other for the interface position. Then by assuming the position a function of the square root of time enabled him to retrieve the Stefan-Nuemann solution. Lightfoot's method has been extended to the solution of two dimensional Stefan problems involving phase change in an infinite wedge [16]. To avoid a tedious iteration in two dimensions, the interface position was taken to be a hyperbolic function and a correction term was introduced into the solution to account for the property variation in different phase regions. Indeed, Lightfoot's method has been limited to phase change of equal properties. This limitation has been lifted by Kolodner [17] by using double source and sink at the phase-change interface.

Integrodifferential equations of Volterra or Fredholm type can also be derived in the solution of the Stefan problems by a Fourier transform [18-19]. This occurs when the transformed equations are inverted in the solution. In general, the integral equation method is useful in providing an exact solution in integral form. However, the integral equations must still be solved numerically for a solution.

One of the unique features of the Stefan problems is the occurrence of multiple phase regions whose domain changes continuously with time. The boundary position of the domain is also sought as a part of the solution. Boley [20] introduced an

embedding method, in which the time-variant domains are embedded in a large fixed domain for the solution of the problem. This introduces an unknown heat flux at the fixed boundary, and this flux must be solved together with the moving boundary position to complete the solution. Boley's embedding method has been extended to the solution of multidimensional problems without internal heat generation [21]. It should be noted that all the moving boundary problems solved in this dissertation also work in a fixed domain; however, a source-and-sink method will be employed which is more convenient to use in the solution of the combination of Stefan and ablation problems.

Stefan problems can also be solved by an asymptotic expansion [22-29]. In this method, a quasi-steady solution is derived by dropping the unsteady temperature terms in the governing equations to find a long-time solution. A quasi-stationary solution is also derived by dropping the interface velocity terms in one of the governing equations and the interface flux condition to establish a short-time solution. Asymptotic expansions of the temperature and the moving boundary position are then constructed by using these solutions for limits [26].

Asymptotic expansions work best in the solution of Stefan problems with nonlinear conditions and convective motions of the melt [22]. Basically a perturbation technique, it offers insight into the physics of the problem. However, the method is very tedious mathematically; even the determination of the first-order terms in the temperature expansion has proven to be an intractable task [26].

Higher-order terms become increasingly difficult to obtain, a distinct drawback of the method.

Coordinate transformation offers another approach to the solution of the Stefan problems [30]. Somewhat similar to the embedding method, the transformation works in a fixed domain without the need of introduction of any new unknowns such as the boundary heat flux in the embedding method. In fact, the time-variant domain is mapped into an invariant domain in the transformation method; the moving boundary is thus immobilized in the solution.

It is noted that use of the transformation does not by itself solve the Stefan problem. It only provides the convenience of working in a fixed domain. The coordinate transformation has thus been used together with other methods for the improvement of accuracy and also in the numerical methods for facilitation of solution [31-37].

Like the Stefan problems briefly reviewed above, ablation problems also fall into the general category of the moving boundary problems. However, the ablation problem is more complex than the Stefan problem because in the latter the problem can be solved in a fixed domain, whereas in the former the domain is continuously changing with time. There have been numerous efforts developed for the solution of the Stefan problems; only very limited work, however, has been devoted to the solution of the ablation problems, which will now be reviewed as follows.

Solution of the ablation problems dates back to Landau [38] who first used the coordinate transformation to change the variable domain encountered in the ablation problem to a fixed domain. An

exact, quasi-steady state solution for the case of a semi-infinite medium imposed with the constant heat flux condition has been given in his paper. Also studied is the ablation for the semi-infinite medium after dropping the steady-state assumption. Using Laplace transformation, Landau was able to derive the pre-melt solution for a time-variant heat flux imposed on the fixed boundary. However, the ablation solution was only derived for the constant heat flux condition. Specifically, two limiting cases were examined that include (i) Stefan number approaching zero (negligible heat capacity) and (ii) Stefan number approaching infinity (negligible latent heat). For the case of the Stefan number of the order of unity, Landau employed a finite difference method to solve the ablation problem with results presented graphically.

Landau's method has also been used by Rogerson and Chayt [39] to find the exact melt-through time for ablation of a slab imposed with a constant heat flux condition on one side and an insulated condition on the other side. Rogerson integrated the heat conduction equation and showed the results to be independent of the thermal properties of the material engaged in phase change.

The integral approach has been used in the solution of the phase-change problems [9,40,41]. Essentially a method of weighted residuals, the method was first introduced by von Karman and Pohlhausen in the approximate solution of boundary layer equations. The heat integral approach is simple to use; it also provides reasonably accurate results. However, it is somewhat handicapped in a detailed analysis of the temperature field. Specifically, the accuracy of the temperature is limited by the form of the profile

initially chosen for analysis. Also the approximation can not be systematically improved for accuracy.

The moment method proposed by Zien [41] carries promise of improvement over the classical heat integral method. The moment method has been used to solve one-dimensional ablation imposed with time-dependent heat-flux conditions. Both pre-melt and ablation solutions were derived. Again, the heat balance integral was used, and the integral of the original heat equation was carried out after multiplying the integrand by powers of temperature. A temperature profile was chosen and substituted into the integrated version of the heat conduction equation. The heat balance integral based on the approximate temperature profile was then used as the expression for the boundary heat flux. Although the method appears to work for the general case of the time-variant heat flux condition, it is expected that the choice of an approximate temperature profile that works for the nonmonotonic heat flux may not be applicable to the case of more general time-variant conditions.

The ablation problems can also be solved by a variational approach. Biot and Agrawal applied the variational analysis and Lagrangian thermodynamics to the solution of ablation problems with variable thermal properties [42]. They considered one-dimensional heat transfer in a semi-infinite cylinder imposed with a constant heat flux condition. Both pre-ablation and ablation stages were solved. In the procedure, the governing equations were transformed and the Lagrangian heat-flow equation was derived which provided a relationship between the surface velocity and the heat penetration depth. Another relation between these two quantities was also

found by using the energy equation. These equations were then solved simultaneously for the heat penetration depth and the velocity of the ablated surface.

The variational method has been applied for approximate analysis of ablation of a semi-infinite solid subject to convective and radiative heating [43,44]. Solutions were obtained in closed form for both the pre-melt and melt-removal heating regimes. In these studies, a cubic temperature profile was taken. The surface temperature, the thermal penetration depth, and the depth of the melt removal were treated as unknown and were determined as functions of time.

The ablation problem has been solved numerically. Recently Blackwell [45] has employed the exponential differencing method to solve an ablation problem in one-dimension. He proposed the use of a moving coordinate system which was attached to the ablated surface. Then, by invoking the use of a finite control-volume approach, the element matrices could be defined for conduction, convection of the moving grid, and energy storage. In this method, all elements and control volumes were moving at a uniform velocity with the exception of the last ablating element and control volume. The node point at the moving interface was fixed in space. As such, the last ablating element had one moving boundary and one fixed boundary. The method has been applied to the solution of a steady-state ablation problem for which an exact solution was available. Comparisons were also made with central differencing for the conduction terms and upwind differencing for the convective terms; the exponential schemes appear to be better numerically.

Ablation problems have also been solved by a finite element method with deforming spatial grids [46]. In this method, the classical finite-element equations are transformed to account for the continuous deformation of the grid for a precise localization of the ablated surface. This is done at the expense of chores of construction of an additional convective matrix.

Recently, ablation has been applied to model intense heating such as laser beam. Masters [47] used the finite difference method to solve the ablation of a one dimensional slab imposed with an intense uniform heat flux and analyzed the effect of melt on the temperature distribution during the heat pulse. The steady-state solutions were found for the velocity of the surface recession and the temperature history during ablation. Abakian and Modest [48] studied the ablation due to a continuous-wave laser beam irradiating on a moving semi-infinite and semitransparent solid. Using an integral method, they were able to derive a set of nonlinear partial differential equations which were solved numerically for the groove depth and shape due to ablation. They also considered the ablation of a moving slab caused by irradiation from continuous-wave and pulsed laser beams and derived a solution for the temperature distribution [49]. The laser has also been used in material processing as investigated by Dabby and Paek [50]. In their work, vaporization occurred at the surface; however, below the surface, the material was heated by absorption of the laser radiation, which might reach a temperature higher than that for vaporization. Explosion may thus take place, which provides a means for material removal in the drilling process.

The review given above is strictly for the solution of regular heat conduction problems. In these problems, the system geometry, governing equation, initial and boundary conditions are fully specified, and the problems are solved mainly for the determination of the temperature field. In the inverse problems, however, the roles of known and unknown quantities are exchanged. There have been abundant studies documented in the literature for the solution of the regular problems; not much work, however, has been devoted to the solution of inverse problems, and for those that have been solved and documented in the literature, nearly all of them are for heat conduction without phase change [51-60]. There is a lack of study for inverse problems with phase change.

Most inverse heat conduction problems deal with a situation where an extra temperature is available at one point in the domain, and this temperature is used together with others to find the condition imposed on the boundary. Stolz [51] first solved such a problem numerically. In his work, the inverse problem was formulated as though it were a direct problem. Since a linear heat conduction problem was solved, he was able to use the superposition principle. An integral equation was derived for the unknown surface condition and was solved by numerical inversion. The method was found to be inefficient if the time steps used were too small. Yet, a small time step must still be used for an accurate solution. Beck [52] was able to improve this method by using a procedure that involved minimization of the sum of the squared difference between the actual and the calculated temperatures at the location where the temperature data were given. Burggraf [53] developed an exact

series solution to a one-dimensional inverse problem with the lump capacitance approximation serving as the leading term. The temperature and heat flux histories were provided at an interior point. Approximate results were found if discrete or experimental data were used for input in solution. Beck [54] moved one step further by solving an inverse problem in which the material properties were treated as functions of temperature; thus the problem solved becomes nonlinear. A two-dimensional inverse finite element solver has also been reported in the literature [55]. It can be used for the solution of the heat flux imposed on the surface of nuclear fuel rod with the use of the interior temperature measurements for input.

Only a handful of studies are found for the solution of inverse heat transfer with phase change. Macqueene et al. [61] proposed an inverse finite element method to determine the efficiency of an arc welding process. In their method, the latent heat of fusion is taken into consideration by the variation of the elemental specific heat when the average temperature of the element reaches the melting point. Conduction in both the solid and liquid regions was accounted for. Katz and Rubinsky [62] proposed a front-tracking finite-element method for the solution of one-dimensional inverse Stefan problems. His method was applied for the determination of the position of the solid-liquid interface and the transient temperature distribution in the solid region during stationary arc welding.

An inverse method was also used by Landram [63] for the analysis of the interface position and the energy transport

mechanisms during welding. The vaporization energy loss was found to be important during the motion of the solid-liquid interface. The interface shape, being close to hemispherical, gives clear indication of a nearly one-dimensional radial symmetry for the heat transfer.

An analytic solution to inverse Stefan problems in Cartesian and spherical geometries was provided by Rubinsky and Shitzer [64]. In their analysis, the inverse Stefan problem was characterized by two boundary conditions at the moving front. This is in sharp contrast to the technique developed in this dissertation (see Chapter 3) which needs only one condition at the interface to solve for the boundary condition. In their work, the medium was initially at the phase change temperature. An integral equation was then derived by integrating the governing equation, which was, in turn, solved by the method of analytic iteration in which the first guessed solution was taken to be the long-time solution to the problem. Series solution was then developed by induction following a number of iterations.

A boundary element analysis with constant elements has been developed by Zabaras et al. for the solution of a one-dimensional inverse solidification problem [1]. They used the sensitivity analysis developed by Beck [52-54] and Burggraf [53] for inverse heat transfer solution. Using an integral formulation, they were able to solve two separate inverse Stefan problems, one in the solid region and the other in the liquid region. Thus similar to the ones given by Rubinsky and Shitzer, two conditions must be provided at the freezing front, also an inefficient method.

It should be mentioned that a source-and-sink method has recently been developed in the heat transfer literature, which is particularly suited for the solution of regular and inverse Stefan problems. In a series of papers, Hsieh and his associates were able to apply this method to the solution of regular and inverse one- and two-phase melting and solidification problems for medium with and without subcooling and superheating and imposed with constant and monotonic temperature and heat flux conditions [2,3]. The method has been applied to the solution of phase change imposed with cyclic conditions [4,5,65]. Two inverse solution techniques have also been developed with the problem formulated with a source-and-sink method as shown in reference 3. The method has shown to be closely related to the boundary element method as reported by Hsieh et al. [66,67]. The present study is a further extension of these works.

CHAPTER III
SOLUTION OF INVERSE STEFAN PROBLEMS
BY A SOURCE-AND-SINK METHOD

This chapter presents the development of a source-and-sink method to solve inverse Stefan problems. In these problems, the conditions are specified at the moving rather than the fixed boundary. Typically, the temporal location of the interface between the phases is given and this location is used together with others to determine the boundary temperature or heat flux that is required to provide for this interface motion.

In what follows in this chapter, the motivation for this study will be given first. It is followed by a general analysis that is designed for the solution of the inverse Stefan problems in three dimensions. This analysis is then used in the solution of one-dimensional inverse problem examples. Finally, results for these examples are provided and discussed in detail and possible extensions of the method are included to conclude this chapter.

3.1 Motivation

Heat diffusion in a medium with constant properties is governed by the partial differential equation

$$\nabla^2 T(\bar{r}, t) + \frac{u'''(\bar{r}, t)}{k} = \frac{1}{\alpha} \frac{\partial T(\bar{r}, t)}{\partial t}, \quad \begin{array}{l} \bar{r} \in R \\ t > 0 \end{array} \quad (3.1)$$

where all notations have their usual meaning. For this medium, the conditions imposed on the boundary are usually one of the following types

$$T(\bar{r}_i, t) = F_i(\bar{r}_i, t), \quad \bar{r}_i \in B_i \quad (3.2)$$

$$\frac{\partial T(\bar{r}_i, t)}{\partial n_i} = -\frac{G_i(\bar{r}_i, t)}{k_i}, \quad \bar{r}_i \in B_i \quad (3.3)$$

$$\frac{\partial T(\bar{r}_i, t)}{\partial n_i} + \frac{h_i}{k_i} T(\bar{r}_i, t) = \frac{1}{k_i} H_i(\bar{r}_i, t), \quad \bar{r}_i \in B_i \quad (3.4)$$

which represent the familiar Dirichlet, Neumann, and Robin conditions, respectively. In (3.3) and (3.4), n_i denotes an outward drawn normal. Then, with the additional initial condition given as

$$T(\bar{r}, 0) = T_i(\bar{r}) \quad (3.5)$$

the temperature solution can be expressed by means of Green's function as [68]

$$T(\bar{r}, t) = \int_{R'} G(\bar{r}, t | \bar{r}', 0) T_i(\bar{r}') dV' + \frac{\alpha}{k} \int_0^t \int_{R'} G(\bar{r}, t | \bar{r}', \tau) u'''(\bar{r}', \tau) dV' d\tau + \sum_i \left\{ \right\} \quad (3.6)$$

Here, the braced term is used to account for the three boundary conditions given earlier. Their expressions are listed in Table 3.1.

A distinct feature is found in the Green's function method above--the effects of the initial condition, heat generation (or destruction), and boundary conditions are embodied respectively in

Table 3.1 Expressions to account for effects of boundary conditions

BOUNDARY CONDITION	{ } EXPRESSION
$T(\overline{r}'_i, t) = F_i(\overline{r}'_i, t)$	$-\alpha \int_0^t \int_{s'_i} \frac{\partial G(\overline{r}, t \overline{r}'_i, \tau)}{\partial n'_i} F_i(\overline{r}'_i, \tau) dS'_i d\tau$
$\frac{\partial T(\overline{r}'_i, t)}{\partial n_i} = \frac{-G_i(\overline{r}'_i, t)}{k_i}$	$-\alpha \int_0^t \int_{s'_i} G(\overline{r}, t \overline{r}'_i, \tau) \frac{G_i(\overline{r}'_i, \tau)}{k_i} dS'_i d\tau$
$\frac{\partial T(\overline{r}'_i, t)}{\partial n_i} + \frac{h_i}{k_i} T(\overline{r}'_i, t) = \frac{1}{k_i} H_i(\overline{r}'_i, t)$	$\alpha \int_0^t \int_{s'_i} G(\overline{r}, t \overline{r}'_i, \tau) \frac{H_i(\overline{r}'_i, \tau)}{k_i} dS'_i d\tau$

the first, second, and third terms on the right of equation (3.6). Then, in the case of regular problems, once these conditions are fully specified, the temperature can be easily found. In this effort, the Green's function can be obtained by using the concept of point charges [68] or by solving auxiliary problems [69].

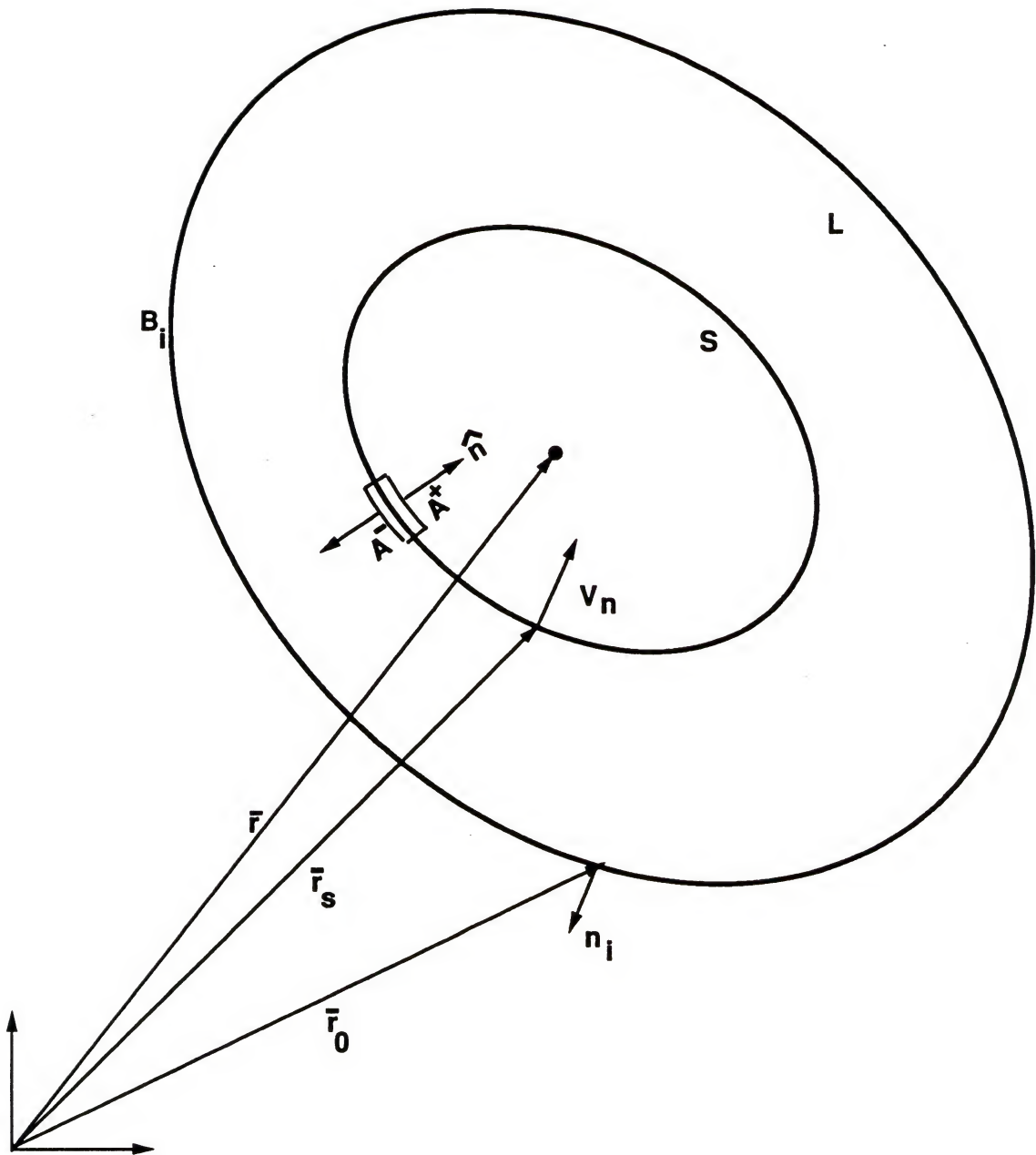
The format of (3.6) turns out to be particularly suited for the solution of the inverse problems. For the inverse problems, the left hand side of this equation can be used to represent the extra temperature that is provided either at the boundary or at interior points. Then this temperature can be used to determine the missing information, such as the initial condition, heat generation, or boundary conditions. In these efforts, the missing quantities can be expanded either in a polynomial or an infinite series, which is substituted into their respective integrals in (3.6). The resulting equation is then solved numerically for the coefficients in the series to complete the solution. This method clearly works for the initial condition and the heat generation. However, for the boundary condition, since it is unknown a priori, one must first assume a particular 'type' of condition that is imposed on the boundary. For convenience, one could use equation (1.2) or (1.3) for this condition. Green's function is then found with this assumed condition. Notice that the boundary condition can be exchanged if necessary as shown in by Hsieh and Shang [70]. That is to say, a Dirichlet condition can be accomplished by means of a Neumann condition and vice versa. The search for the condition is thus not restricted by the type of the conditions assumed. Better yet, an incremental solution approach can also be developed to track

the boundary conditions accurately as will be shown later. The concept above will now be applied for the solution of inverse Stefan problems in which the interface positions are given, and these positions are used to find the boundary conditions that must be imposed to cause the interface motions.

3.2 General Analysis

For the sake of illustration in what follows, the Stefan problems consist of two stages: a pre-melt stage, when heat is added to the surface of a subcooled medium to raise its temperature to the phase-change temperature; and a melting stage, when the medium changes phase and the melting starts at the surface and the interface moves inward with time. It is assumed that the properties for different phases are constant and of equal value. The medium has a distinct melting temperature; that is, no mushy zone in the medium. Convection is negligible. For generality, the analysis will be developed for the solution of both melting and solidification problems. The analysis can also be extended for the solution of Stefan problems in multiple phases and in a medium with unequal phase properties as will be discussed later. For the moment, the simplified problems will be solved by considering the medium shown in Figure 3.1. The formulation of these problems follows below.

Figure 3.1 System analyzed



Pre-melt Stage

Governing equation--

$$\begin{aligned} \bar{r} &\in (L \cup S) \\ \nabla^2 T_0(\bar{r}, t) &= \frac{1}{\alpha} \frac{\partial T_0(\bar{r}, t)}{\partial t}, \\ t_0 &\geq t > 0 \end{aligned} \tag{3.7}$$

Initial condition--

$$T_0(\bar{r}, 0) = T_i(\bar{r}) \tag{3.8}$$

Melting Stage

Liquid Region:

Governing equation--

$$\begin{aligned} \bar{r} &\in L \\ \nabla^2 T_L(\bar{r}, t) &= \frac{1}{\alpha} \frac{\partial T_L(\bar{r}, t)}{\partial t}, \\ t &> t_0 \end{aligned} \tag{3.9}$$

Solid Region:

Governing equation--

$$\begin{aligned} \bar{r} &\in S \\ \nabla^2 T_S(\bar{r}, t) &= \frac{1}{\alpha} \frac{\partial T_S(\bar{r}, t)}{\partial t}, \\ t &> t_0 \end{aligned} \tag{3.10}$$

Initial condition--

$$T_S(\bar{r}, t_0) = T_0(\bar{r}, t_0) \quad (3.11)$$

Interface Conditions:

$$T_L(\bar{r}_f, t) = T_m = T_S(\bar{r}_f, t) \quad (3.12)$$

$$\frac{\partial T_S(\bar{r}_f, t)}{\partial n} - \frac{\partial T_L(\bar{r}_f, t)}{\partial n} = \frac{\rho L}{k} v_n(t) \quad (3.13)$$

$$v_n(t) = \bar{V} \cdot \hat{n} \quad (3.14)$$

Here \bar{r}_f denotes the interface position and v_n the history of the interface motion. For the inverse Stefan problems of interest in this study, this history is used for the determination of the missing boundary conditions.

The problems as posed can be solved by use of the Green's function method described in the preceding section. However, a direct use of this method would require (3.6) to be applied to two separate regions, liquid and solid, and the solution so obtained may not be as efficient as one desires. A source-and-sink method is thus used [2,4,15]. In this method, the melting interface is taken to be a moving heat-sink front and a freezing interface is taken to be a moving heat-source front. Then, in sharp contrast to conventional methods in which different equations are used to represent the temperatures in different regions, only one equation will be derived. Whether it is in the solid or liquid region is

determined by the position that is assigned in the temperature equation. The solution of the inverse problems can then be simplified with this method. Following this approach, the melting stage is solved by considering an equivalent problem as follows:

Governing equation for the equivalent problem:

$$\begin{aligned} \bar{r} &\in (L \cup S) \\ \nabla^2 T(\bar{r}, t) + \frac{\rho L}{k} v_n(t) \delta(\bar{r} - \bar{r}_f) &= \frac{1}{\alpha} \frac{\partial T(\bar{r}, t)}{\partial t}, \\ t &> t_0 \end{aligned} \quad (3.15)$$

Initial condition for the equivalent problem:

$$T(\bar{r}, t_0) = T_0(\bar{r}, t_0) \quad (3.16)$$

Interface conditions for the equivalent problem:

$$T(\bar{r}_f, t) = T_m, \quad v_n(t) = \bar{V} \cdot \hat{n} \quad (3.17a, b)$$

where $\delta(\bar{r} - \bar{r}_f)$ denotes a Dirac delta function. The signs preceding this function are used for freezing (+) and melting (-).

It can be shown readily that (3.15) reduces to (3.9) and (3.10). Furthermore, by integrating (3.15) across the interface from $\bar{r}_f - e$ to $\bar{r}_f + e$ and forcing e to be zero in a limiting process, equation (3.15) reduces to (3.13). This can be proved by using the pill box at the interface in Figure 3.1. Other equivalences for this stage are apparent.

The equivalent problem can be solved by referring to (3.6) in which the heat generation term is changed to the interface motion term as

$$T(\bar{r}, t) = \int_{L \cup S} G(\bar{r}, t | \bar{r}', t_0) T_i(\bar{r}') dV' \pm \frac{L}{c} \int_{t_0}^t \int_{L \cup S} G(\bar{r}, t | \bar{r}', \tau) v_n(\tau) \delta(\bar{r}' - \bar{r}_f) dV' d\tau + \sum_i \left\{ \right\} \quad (3.18)$$

where the plus sign is used for freezing and minus sign for melting. Finally, the boundary conditions can be found by setting \bar{r} in this equation to the interface position, \bar{r}_f , and $T(\bar{r}_f, t)$ to the melting temperature, T_m , as

$$T_m = \int_{L \cup S} G(\bar{r}_f, t | \bar{r}', t_0) T_i(\bar{r}') dV' \pm \frac{L}{c} \int_{t_0}^t \int_{L \cup S} G(\bar{r}_f, t | \bar{r}', \tau) v_n(\tau) \delta(\bar{r}' - \bar{r}_f) dV' d\tau + \sum_i \left\{ \right\} \quad (3.19)$$

The missing boundary conditions can then be found by solving them implicitly. In this effort, the time when melting starts (t_0) can be determined by solving the pre-melt problem, whose solution can again be taken to be the special case of (3.6) in which the heat generation term is zero. Solution in this stage is thus elementary.

3.3 Example Problems

The analysis above is now used to solve example problems which are in semi-infinite domain in which the interface motion is given. For the sake of generality, the analysis will be developed for determining either the Dirichlet condition or the Neumann condition that is imposed on the boundary at $x=0$, and the interface motion may

be the result of either melting or solidification in a one- and two-phase medium. Also for illustration purposes, the heat flow is one dimensional, the initial temperature being uniform. Extensions to more dimensions are provided in Appendix A.

For the problem given, Green's function can be found to be

$$G(x,t | x',\tau) = \frac{1}{2\sqrt{\pi\alpha(t-\tau)}} \left[\exp\left(-\frac{(x-x')^2}{4\alpha(t-\tau)}\right) \pm \exp\left(-\frac{(x+x')^2}{4\alpha(t-\tau)}\right) \right] \quad (3.20)$$

where the plus and minus signs are to be used when the flux and temperature condition is assumed to appear at the boundary, respectively. The temperature can then be obtained by using (3.18), which is recast in a general format as

$$\frac{T(x,t)}{T_m} = \frac{T_0(x,t)}{T_m} \pm \frac{\hat{H}(t-t_0)}{Ste} \int_0^{t-t_0} \frac{dR(\tau+t_0)}{d\tau} G(x,t | R(\tau+t_0),\tau) d\tau \quad (3.21)$$

where all temperatures, including T_m , are measured in excess of the initial temperature, and

$$T_0(x,t) = \sqrt{\frac{\alpha}{\pi}} \int_0^t \frac{E(s)}{(t-s)^{1/2}} \exp\left[-\frac{x^2}{4\alpha(t-s)}\right] ds \quad (3.22)$$

in which

$$E(s) = \begin{cases} \frac{x}{2\alpha} \frac{F(s)}{t-s} \\ \frac{1}{k} G(s) \end{cases} \quad (3.23a,b)$$

Here $F(s)$ and $G(s)$ denote the assumed temperature and heat flux condition, respectively. Also

$$\hat{H}(t-t_0) = \begin{cases} 1 & t > t_0 \\ 0 & t \leq t_0 \end{cases} \quad \text{for} \quad (3.24)$$

$$\text{Ste} = \frac{cT_m}{L} \quad (3.25)$$

where Ste is known as the Stefan number. With the use of the circumflexed Heaviside function given by (3.24), equation (3.21) holds for all time and for both one- and two-phase problems.

Equation (3.21) can be used to determine the unknown boundary condition by invoking use of the condition at the interface as

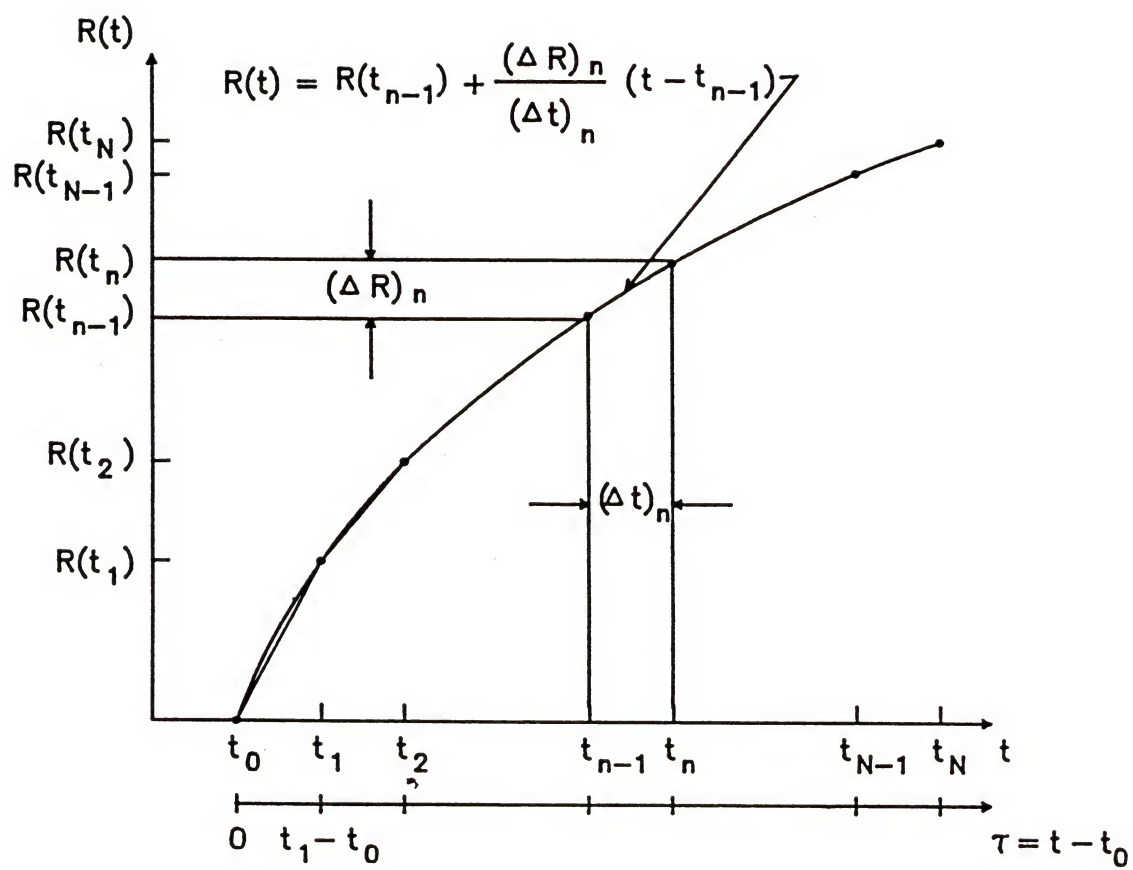
$$\pm \left[1 - \frac{T_0(R(t), t)}{T_m} \right] = \frac{\hat{H}(t-t_0)}{\text{Ste}} \int_0^{t-t_0} \frac{dR(\tau+t_0)}{d\tau} G(R(t), t-t_0 | R(\tau+t_0), \tau) d\tau \quad (3.26)$$

where the plus and minus signs on the left hand side are to be used for freezing and melting, respectively.

3.4 Numerical Solution

A local linearization can be used to solve (3.26) numerically. In this effort, the entire time range is divided into small increments in which the interface position is taken to be linear; see Figure 3.2. Then the dR/dt can be taken out of the integral for each increment, and the convolution integral written as a summation as

Figure 3.2 Linearization of solid-liquid interface
position curves for the numerical solution



$$\pm \frac{T_0(R(t_N), t_N)}{T_m} = \pm 1 - \frac{\hat{H}(t_N - t_0)}{Ste} \sum_{n=1}^N \left[\frac{dR(t_n^-)}{dt} \int_{t_{n-1}-t_0}^{t_n-t_0} G(R(t_N), t_N - t_0 | R(\tau+t_0), \tau) d\tau \right] \quad (3.27)$$

Here the signs are to be selected following the statement below (3.26). For the inverse problems, the right hand side can be evaluated with the input of the interface motion data. As for the left hand side, if the boundary conditions are represented by a power series, then the number of terms on this side must be equal to the number of the terms that are taken in the series. In practice, equation (3.27) can be written by means of matrix elements for a melting problem for the series method as

$$\sum_{n=1}^N b_{mn} a_n = c_m + \sum_{n=1}^N d_{mn} e_n \quad (3.28)$$

where $m=N$; $t_N > t_0$; $N=1, 2, \dots, N$; and

$$b_{mn} = \begin{cases} 0 & \text{for } m < n \\ \frac{R(t_m)}{2\alpha} \int_0^{t_m} \frac{s^{n-1}}{(t_m - s)^{3/2}} \exp\left[-\frac{R(t_m)^2}{4\alpha(t_m - s)}\right] ds & \text{for } m \geq n; F(s) = \sum_{n=1}^N a_n s^{n-1} \\ \frac{1}{k} \int_0^{t_m} \frac{s^{n-1}}{(t_m - s)^{1/2}} \exp\left[-\frac{R(t_m)^2}{4\alpha(t_m - s)}\right] ds & \text{for } m \geq n; G(s) = \sum_{n=1}^N a_n s^{n-1} \end{cases} \quad (3.29a,b,c)$$

$$c_m = \sqrt{\frac{\pi}{\alpha}} T_m \quad (3.30)$$

$$d_{mn} = \begin{cases} 0 & \text{for } m < n \\ \frac{L}{c\sqrt{\alpha}} \int_{t_{n-1}-t_0}^{t_n-t_0} G(R(t_m), t_m - t_0 | R(\tau + t_0), \tau) d\tau & \text{for } m \geq n \end{cases} \quad (3.31a,b)$$

$$e_n = \frac{dR(t_n^-)}{dt} \quad (3.32)$$

Notice that the integrals in these equations can be performed in closed form if the missing conditions are expressed in the power series as shown in (3.29) or in terms of a Fourier series. The coefficient matrices B and D are of a lower triangular structure, a characteristic permitting the solution of (3.28) to be carried out simply by using forward substitution, a simple numerical procedure.

The boundary conditions can also be determined by use of an incremental approach. In this method, (3.27) is recast as

$$\begin{aligned} & \pm \int_{t_{N-1}}^{t_N} \frac{E(s)}{(t_N - s)^{1/2}} \exp\left[-\frac{R(t_N)^2}{4\alpha(t_N - s)}\right] ds = \\ & \sqrt{\frac{\pi}{\alpha}} T_m \left\{ \pm 1 - \frac{\widehat{H}(t_N - t_0)}{\text{Ste}} \sum_{n=1}^N \left[\frac{dR(t_n^-)}{dt} \int_{t_{n-1}-t_0}^{t_n-t_0} G(R(t_N), t_N - t_0 | R(\tau + t_0), \tau) d\tau \right] \right\} - \\ & \left\{ \pm \sum_{n=1}^{N-1} \int_{t_{n-1}}^{t_n} \frac{E(s)}{(t_N - s)^{1/2}} \exp\left[-\frac{R(t_N)^2}{4\alpha(t_N - s)}\right] ds \right\} \end{aligned} \quad (3.33)$$

where t_0 is set to zero all the time in the integral on the left and

the last integral on the right. In practice, equation (3.33) can be recast as

$$\begin{pmatrix} F(t_N) \\ G(t_N) \end{pmatrix} = \begin{pmatrix} P_{N,F} \\ P_{N,G} \end{pmatrix}^{-1} \left\{ \sqrt{\frac{\pi}{\alpha}} T_m + \sum_{n=1}^N d_{Nn} e_n - \sum_{n=1}^{N-1} \begin{pmatrix} F(t_n) P_{n,F} \\ G(t_n) P_{n,G} \end{pmatrix} \right\} \quad (3.34 \text{ a,b})$$

where $t_N > t_0$; $N=1,2,\dots,N$; d_{Nn} can be obtained by referring to (3.31); and

$$\begin{pmatrix} P_{n,F} \\ P_{n,G} \end{pmatrix} = \begin{cases} \frac{R(t_N)}{2\alpha} \int_{t_{n-1}}^{t_n} \frac{1}{(t_N-s)^{3/2}} \exp\left[-\frac{R(t_N)^2}{4\alpha(t_N-s)}\right] ds \\ \frac{1}{k} \int_{t_{n-1}}^{t_n} \frac{1}{(t_N-s)^{1/2}} \exp\left[-\frac{R(t_N)^2}{4\alpha(t_N-s)}\right] ds \end{cases} \quad (3.35 \text{ a,b})$$

In (3.34), the summation vanishes when $N-1=0$. Then starting from $N=1, 2$, and so on, the conditions can be evaluated incrementally. In this effort, the F and G values found from the previous time steps are used as the input in the computation of the right hand side of (3.34), which immediately gives the F and G values at the succeeding new time step. This continues until the desired time is reached. The algorithms can be developed for a rapid solution of the conditions. Computer programs developed for the solution of inverse Stefan problems given in this chapter are provided in Appendices B through E.

3.5 Critique of the Method

Using Green's functions, the present method is limited to the solution of problems whose Green's function can be obtained analytically. This excludes those problems whose boundary conditions and governing equations are nonlinear and such nonlinearity can not be resolved by using transformation (e.g., Kirchhoff transformation). Yet with the use of Green's function, it projects an impression of being related to the boundary element method that has been undergoing through development in recent years [64]. Yet, they are functionally different. In boundary element method, the boundary element equations are written separately for the liquid and solid regions, whereas in the present method, only one equation (3.27) is derived. The number of the equations to be solved in the present method is thus reduced by half, which is particularly true in the solution by the incremental approach described earlier.

More important, in the boundary element method, the problems cannot be solved without the information for the heat fluxes that appear on both sides of the interface [64,1]. Such fluxes, however, are unnecessary in the present work. Instead, equation (3.26) embodies conditions (3.12) and (3.13) in the form of a single integrodifferential equation. There is no need for the satisfaction of the flux conditions; as a result, the present method is more effective because it requires less information for input.

As will be shown in the next section, the present method is also accurate. In fact, such accuracy is not unexpected because equation (3.26) is exact. The only approximation in the analysis is

the local linearization which has been applied to the interface motion and the boundary condition; see (3.33) and (3.35). In fact, in the present analysis, they have been approximated by using constant elements. The analysis can thus be improved for accuracy by using higher order elements, such as linear and quadratic elements as in references [59,71].

3.6 Results and Discussion

For the numerical experiments performed in this study, the interface motion data are taken from reference 5. Aluminum will be used for tests; its properties are given in Table 3.2. The inverse solution techniques developed in this chapter are used to solve eight example problems of which four having exact solutions. The retrieved conditions for these examples can thus be compared with the exact solutions for error. In these four examples, the interfaces move as a function of square root of time, and the interface motion data are used to retrieve the constant temperature conditions that appear on the surface. This problem is known as the Stefan-Neumann problem, and Table 3.2 provides a summary of the conditions tested in the examples.

The first example deals with a general two-phase Stefan-Neumann problem; the medium is initially subcooled to 300 K, which is lower than the melting temperature (see Table 3.3). For this example, the temperature at the boundary is unknown; an unknown temperature is thus assumed to appear at the boundary, and the interface motion data are used to find this temperature. The results are listed in Table 3.4, where two sets of results are

Table 3.2 Properties of aluminum

Melting temperature, T_m (K)	932
Vaporization temperature, T_v (K)	2,543
Heat of fusion, L_f (J/kg)	389,600
Heat of vaporization, L_v (J/kg)	9,462,000
Thermal conductivity, k (W/m K)	200
Density, ρ (kg/m ³)	2,710
Specific heat, c (J/kg K)	1200

Table 3.3 Conditions tested in eight examples

PROBLEM DESCRIPTION		INPUT DATA	TRUE CONDITION $F(t):K; G(t):W/m^2$	TYPE OF BOUNDARY CONDITION	
				ASSUMED	SOLVED
1	two-phase $T_i=300K < T_m$	$R(t) \sim \sqrt{t}$	$F(t)=1000$	temperature	temperature
1	two-phase $T_i=300K < T_m$	$R(t) \sim \sqrt{t}$	$F(t)=1000$	heat flux	temperature
3	one-phase $T_i=932K = T_m$	$R(t) \sim \sqrt{t}$	$F(t)=1000$	temperature	temperature
4	one-phase $T_i=932K = T_m$	$R(t) \sim \sqrt{t}$	$F(t)=1000$	heat flux	temperature
5	one-phase $T_i=932K = T_m$	reference [5]	$F(t)=1000+5t$	temperature	temperature
6	two-phase $T_i=300K < T_m$	reference [5]	$G(t)=6.388034 \times 10^6 + 2.82752 \times 10^5 t$	heat flux	heat flux
7	two-phase $T_i=300K < T_m$	reference [5]	$F(t)=932+40t$	temperature	temperature
8	one-phase $T_i=932K = T_m$	reference [5]	$G(t)=3 \times 10^6 + 5 \times 10^4 t^2$	heat flux	heat flux

Table 3.4 Comparison between true and retrieved conditions for first Stefan-Neumann example solved for boundary temperature

Time (sec)	Boundary Conditions, T (K)			
	Imposed True Condition	Retrieved		
		Series Method		Incremental Method
		N=3	N=4	
1	1000.00	1003.22	1003.17	1003.49
2	1000.00	1002.97	1002.87	1000.41
3	1000.00	1002.72	1002.58	1000.20
4	1000.00	1002.48	1002.30	1000.11
5	1000.00	1002.25	1002.04	1000.03
6	1000.00	1002.03	1001.79	1000.05
7	1000.00	1001.82	1001.56	1000.04
8	1000.00	1001.61	1001.34	1000.02
9	1000.00	1001.42	1001.14	1000.68
10	1000.00	1001.23	1000.95	999.88
11	1000.00	1001.06	1000.78	999.97
12	1000.00	1000.89	1000.63	999.99
13	1000.00	1000.73	1000.50	999.99
14	1000.00	1000.58	1000.38	999.99
15	1000.00	1000.43	1000.28	999.99
16	1000.00	1000.30	1000.20	999.98
17	1000.00	1000.17	1000.14	999.98
18	1000.00	1000.06	1000.09	999.98
19	1000.00	999.95	1000.07	999.90
20	1000.00	999.85	1000.07	999.94
<div> $F(s) = \sum_{n=1}^N a_n s^{n-1}$ <div> For N=3, $a_1=1003.490405$ $a_2= -0.268732$ $a_3= 4.350448 \times 10^{-3}$ </div> <div> For N=4, $a_1=1003.490405$ $a_2= -0.322478$ $a_3= 6.264645 \times 10^{-3}$ $a_4= 6.552981 \times 10^{-5}$ </div> </div>				

given--one for the series solution method and the other for the incremental solution method.

In the series solution method, a power series of degree $N-1$ is used to represent the temperature (see (3.29b)), and two values of N are tested. In this method, the interface position data at equal time intervals are used for input. Thus, for example, for $N=3$, R data at times equal to 0, 6, 12, and 18 seconds are used to determine the series, which is, in turn, used to generate the temperature values listed for 20 time steps in the table. Comparing the retrieved temperatures at the boundary by both series with the true condition (exact temperature) listed to the left indicates they are in good agreement. In this case, the coefficients found for these series are listed at the bottom of the table. From the temperature values tabulated, the results for the low power series appear to be as good as those of the high power series, and such trend persists even with the test of a higher power series of degree 10 (results not shown). This gives the indication that the temperatures have been converged. As for the incremental solution method, the results are also good; errors are of the order of 10^{-3} % at large time. For the incremental method, the boundary conditions are found at exactly the same times when the interface positions are given. The time step for the solution is thus identical to that for the interface data input. Also notice that the convergence and stability that are normally encountered in the conventional finite difference methods are nonexistent in the present incremental solution of integral equations. Also as in the case of the series solution method, the incremental solution results are generally

better at large time than small time, which will be further discussed later.

In the example above, a temperature condition is imposed, and the same 'type' of condition is assumed in the process of the inverse solution. Since the type of the condition that is imposed on the boundary is unknown a priori, it may well be the heat flux condition that one assumes, and the question to be addressed now is whether it is still possible to retrieve the temperature condition via the assumed flux condition. Use will now be made of equation (3.21) to determine this temperature condition once the boundary flux condition is found, and the results are listed in Table 3.5. As shown in the table, the incremental solution results are still good but the series solution results are not as accurate as those listed in Table 3.4. This is certainly a result of the errors being accumulated first in the evaluation of the heat flux next in the evaluation of the temperature using the previously determined heat flux. While this example serves well to illustrate that the conditions are still exchangeable, such a two-step solution of the condition may lead to large errors, particularly in the series solution method, and should thus be avoided in practice. In fact, according to experience, the computer time saved in this two-step approach of solving flux then temperature is insignificant as compared with that in the separate, one-step approach of direct evaluation of the flux and temperature.

A slight modification is made in the next two examples: this time the medium is not subcooled, the initial temperature being equal to the melting temperature of the medium (see examples 3 and 4

Table 3.5 Comparison between true and retrieved conditions for second Stefan-Neumann example solved for boundary flux then temperature

Time (sec)	Boundary Conditions, T (K)			
	Imposed True Condition	Retrieved		
		Series Method		Incremental Method
		N=4	N=5	
1	1000.00	585.89	620.85	1023.06
2	1000.00	715.58	758.46	1007.71
3	1000.00	799.99	844.90	1003.60
4	1000.00	862.06	905.50	1000.88
5	1000.00	909.11	948.70	1000.04
6	1000.00	945.07	979.11	999.68
7	1000.00	972.34	999.67	999.52
8	1000.00	992.57	1012.44	999.44
9	1000.00	1006.96	1019.03	998.11
10	1000.00	1016.46	1020.76	998.30
11	1000.00	1021.85	1018.76	998.37
12	1000.00	1023.76	1014.02	998.44
13	1000.00	1022.76	1007.48	998.51
14	1000.00	1019.34	999.99	998.59
15	1000.00	1013.95	992.40	998.66
16	1000.00	1007.01	985.51	999.45
17	1000.00	998.89	980.13	999.16
18	1000.00	989.05	977.04	999.11
19	1000.00	980.54	977.04	999.07
20	1000.00	971.00	980.93	997.15
$G(s) = \sum_{n=1}^N a_n s^{n-1}$ <div> For N=4, $a_1=8001187.956002$ $a_2= -387201.858263$ $a_3= 1581.702473$ $a_4= 148.390536$ For N=5, $a_1=8945600.085189$ $a_2= -541131.047558$ $a_3= 2763.120508$ $a_4= 324.034498$ $a_5= 3.817008$ </div>				

in Table 3.3). The Stefan-Neumann problems solved thus become a one-phase problem. It will be tested that the results are unaffected by the change to the one-phase problem. Again, the same series of tests are made and the results are listed in Tables 3.6 and 3.7. Again, the one-step solution results are good (Table 3.6). The two- step solution results are poorer (Table 3.7), further reinforcing the recommendation made earlier in the testing of the two-phase problems.

Having satisfactorily completed testing of the Stefan-Neumann problems, attention is now directed to the solution of inverse Stefan problems whose interface motions must be met by imposing the time-variant temperature and flux conditions. There are no exact solutions for these problems, and the interface motion data are taken from Choi [5] who have solved the regular (forward) version of the problems with great accuracy. The interface position data are then used to retrieve the boundary conditions and the results are listed in Tables 3.8 through 3.11. Tables 3.8 and 3.9 give the results for the direct retrieval of the linear temperature and heat flux conditions

$$F(t) = 1000 + 5t \quad (3.36)$$

$$G(t) = 6.388034 \times 10^6 + 2.82752 \times 10^5 t \quad (3.37)$$

while Tables 3.10 and 3.11 give the results for the direct retrieval of the linear temperature and quadratic heat flux conditions

Table 3.6 Comparison between true and retrieved conditions for third Stefan-Neumann example solved for boundary temperature

Time (sec)	Boundary Conditions, T (K)			
	Imposed True Condition	Retrieved		
		Series Method		Incremental Method
		N=3	N=4	
1	1000.00	1033.83	1033.12	1037.49
2	1000.00	1030.31	1028.96	998.22
3	1000.00	1026.93	1025.01	998.58
4	1000.00	1023.69	1021.29	998.85
5	1000.00	1020.59	1017.80	999.05
6	1000.00	1017.63	1014.55	999.20
7	1000.00	1014.81	1011.55	999.24
8	1000.00	1012.13	1008.81	999.38
9	1000.00	1009.59	1006.33	999.44
10	1000.00	1007.19	1004.13	999.49
11	1000.00	1004.93	1002.21	999.53
12	1000.00	1002.81	1000.57	999.59
13	1000.00	1000.82	999.23	999.60
14	1000.00	998.98	998.20	999.65
15	1000.00	997.28	997.47	999.65
16	1000.00	995.71	997.07	999.69
17	1000.00	994.29	997.00	999.69
18	1000.00	993.00	997.26	999.73
19	1000.00	991.86	997.87	999.71
20	1000.00	990.85	998.82	999.78
$F(s) = \sum_{n=1}^N a_n s^{n-1}$ <div> <div>For N=3, $a_1=1037.496339$</div> <div>$a_2= -3.728416$</div> <div>$a_3= 6.982696 \times 10^{-2}$</div> <div>For N=4, $a_1=1037.496324$</div> <div>$a_2= -4.474102$</div> <div>$a_3= 1.005513 \times 10^{-1}$</div> <div>$a_4= 1.324365 \times 10^{-3}$</div> </div>				

Table 3.7 Comparison between true and retrieved conditions for fourth Stefan-Neumann example solved for boundary flux then temperature

Time (sec)	Boundary Conditions, T (K)			
	Imposed True Condition	Retrieved		
		Series Method		Incremental Method
		N=4	N=5	
1	1000.00	906.32	917.43	1056.39
2	1000.00	953.87	966.25	996.60
3	1000.00	980.48	991.90	1002.26
4	1000.00	997.67	1006.84	999.33
5	1000.00	1008.95	1015.10	999.70
6	1000.00	1016.02	1018.67	999.45
7	1000.00	1019.89	1018.86	999.16
8	1000.00	1021.28	1016.61	999.22
9	1000.00	1020.73	1012.66	999.09
10	1000.00	1018.66	1007.67	999.23
11	1000.00	1015.44	1002.18	999.05
12	1000.00	1011.39	996.73	999.32
13	1000.00	1006.79	991.80	998.89
14	1000.00	1001.89	987.87	999.72
15	1000.00	996.94	985.38	998.27
16	1000.00	992.16	984.80	1001.19
17	1000.00	987.77	986.56	994.04
18	1000.00	983.98	991.12	1014.19
19	1000.00	980.99	998.92	951.97
20	1000.00	979.00	1010.44	1162.46
$G(s) = \sum_{n=1}^N a_n s^{n-1}$ <div> <div>For N=4, $a_1=2805963.114624$</div> <div>$a_2= -204888.890714$</div> <div>$a_3= 2268.882336$</div> <div>$a_4= 99.892255$</div> <div>For N=5, $a_1=3137161.977178$</div> <div>$a_2= -286340.839757$</div> <div>$a_3= 3963.574559$</div> <div>$a_4= 218.130426$</div> <div>$a_5= 1.467676$</div> </div>				

Table 3.9 Comparison between true and retrieved conditions for sixth inverse example problem solved for heat flux

Time (sec)	Boundary Conditions, q (W/m ²)						
	Imposed True Condition	Retrieved					
		Series Method				Incremental Method	Error %
		N=4	Error %	N=5	Error %		
4.25	7589730.94	8076200.72	6.41	7558675.50	-0.40	7184872.69	-5.33
4.50	7660418.94	8147001.48	6.35	7694176.58	0.44	7371675.43	-3.77
4.75	7731106.94	8206716.05	6.15	7815212.59	1.08	7493717.81	-3.07
5.00	7801794.94	8257180.74	5.83	7920261.67	1.52	7596454.24	-2.63
5.25	7872482.94	8300231.83	5.43	8008672.53	1.73	7689904.72	-2.31
5.50	7943170.94	8337705.60	4.96	8080664.49	1.73	7775533.53	-2.11
5.75	8013858.94	8371438.35	4.46	8137327.48	1.54	7859641.15	-1.92
6.00	8084546.94	8403266.35	3.94	8180622.00	1.19	7940899.93	-1.77
6.25	8155234.94	8435025.90	3.43	8213379.15	0.71	8020420.49	-1.65
6.50	8225922.94	8468553.29	2.94	8239300.64	0.16	8098888.64	-1.54
6.75	8296610.94	8505684.79	2.52	8262958.75	-0.40	8176244.61	-1.45
7.00	8367298.94	8548256.71	2.16	8289796.37	-0.93	8252784.49	-1.36
7.25	8437986.94	8598105.32	1.89	8326126.99	-1.33	8329871.72	-1.28
7.50	8508674.94	8657066.91	1.74	8379134.69	-0.61	8402584.80	-1.25
7.75	8579362.94	8726977.77	1.72	8456874.13	-1.42	8480061.68	-1.15
8.00	8650050.94	8809674.19	1.84	8568270.59	-0.94	8553180.08	-1.12
8.25	8720738.94	8906992.45	2.13	8723119.93	0.02	8628203.50	-1.06
8.50	8791426.94	9020768.84	2.60	8932088.60	1.60	8701314.04	-1.02
8.75	8862114.94	9152839.65	3.28	9206713.65	3.88	8775467.98	-0.98
9.00	8932802.94	9305041.17	4.16	9559402.73	7.01	8847997.32	-0.95

For N=4, $a_1=3397040.84$
 $a_2=2247925.15$
 $a_3=-353114.747$
 $a_4=19587.0623$

For N=5, $a_1=8734725.30$
 $a_2=-3092029.40$
 $a_3=1293821.95$
 $a_4=-188030.644$
 $a_5=9286.34410$

Time (sec)	Boundary Conditions, T (K)						
	Imposed True Condition	Retrieved					
		Series Method				Incremental Method	Error %
		N=3	Error %	N=4	Error %		
1	972.00	976.35	0.44	1020.27	4.97	969.93	-0.21
2	1012.00	1014.88	0.28	1044.41	3.20	1006.51	-0.54
3	1052.00	1053.63	0.15	1071.70	1.87	1042.85	-0.87
4	1092.00	1092.58	0.05	1101.84	0.90	1079.43	-1.15
5	1132.00	1131.74	-0.02	1134.51	0.22	1116.09	-1.40
6	1172.00	1171.11	-0.07	1169.40	-0.22	1153.87	-1.55
7	1212.00	1210.69	-0.10	1206.22	-0.47	1192.15	-1.64
8	1252.00	1250.48	-0.12	1244.64	-0.58	1230.89	-1.69
9	1292.00	1290.48	-0.11	1284.37	-0.59	1270.11	-1.69
10	1332.00	1330.70	-0.09	1325.09	-0.51	1310.73	-1.60
11	1372.00	1371.12	-0.06	1366.49	-0.40	1348.84	-1.69
12	1412.00	1411.75	-0.01	1408.27	-0.26	1389.93	-1.56
13	1452.00	1452.59	0.04	1450.12	-0.12	1429.95	-1.52
14	1492.00	1493.64	0.11	1491.73	-0.01	1470.27	-1.46
15	1532.00	1534.91	0.19	1532.80	0.05	1510.54	-1.40
16	1572.00	1576.38	0.27	1573.01	0.06	1550.77	-1.35
17	1612.00	1618.06	0.37	1612.05	0.003	1590.93	-1.31
18	1652.00	1659.95	0.48	1649.63	-0.14	1631.04	-1.27
19	1692.00	1702.06	0.59	1685.42	-0.39	1671.39	-1.22
20	1732.00	1744.37	0.71	1719.12	-0.74	1710.13	-1.26

$$F(s) = \sum_{n=1}^N a_n s^{n-1}$$

For N=3, $a_1=938.0379452$
 $a_2= 38.2157864$
 $a_3= 0.1050509$

For N=4, $a_1=999.5873340$
 $a_2= 18.8517864$
 $a_3= 1.8834608$
 $a_4= -0.0513597$

$$F(s) = \sum_{n=1}^N a_n s^{n-1}$$

Table 3.11 Comparison between true and retrieved conditions for eight inverse example problem solved for heat flux

Time (sec)	Boundary Conditions, q (W/m ²)						
	Imposed True Condition	Retrieved				Incremental Method	Error %
		Series Method					
		N=4	Error %	N=5	Error %		
0.25	3003125.00	3007778.76	0.15	3033589.06	1.01	2995616.25	-0.25
0.50	3012500.00	3021875.02	0.31	3035992.94	0.78	3004320.50	-0.27
0.75	3028125.00	3040169.58	0.40	3043865.79	0.52	3019590.53	-0.28
1.00	3050000.00	3062934.56	0.42	3057990.24	0.26	3036905.38	-0.43
1.25	3078125.00	3090442.06	0.40	3078999.09	0.02	3061522.20	-0.54
1.50	3112500.00	3122964.20	0.34	3107375.31	-0.16	3092582.42	-0.64
1.75	3153125.00	3160773.08	0.24	3143452.06	-0.31	3130006.55	-0.73
2.00	3200000.00	3204140.84	0.13	3187412.63	-0.39	3173845.90	-0.82
2.25	3253125.00	3253339.57	0.01	3239290.49	-0.43	3223966.45	-0.90
2.50	3312500.00	3308641.38	-0.12	3298969.30	-0.41	3280396.20	-0.97
2.75	3378125.00	3370318.40	-0.23	3366182.86	-0.35	3343088.04	-1.04
3.00	3450000.00	3438642.74	-0.33	3440515.14	-0.27	3411906.45	-1.10
3.25	3528125.00	3513886.50	-0.40	3521400.30	-0.19	3487162.49	-1.16
3.50	3612500.00	3596321.81	-0.45	3608122.64	-0.12	3568166.70	-1.23
3.75	3703125.00	3686220.76	-0.46	3699816.64	-0.08	3656322.04	-1.26
4.00	3800000.00	3783855.48	-0.42	3795466.96	-0.12	3748732.19	-1.35
4.25	3903125.00	3889498.09	-0.35	3893908.40	-0.23	3851068.35	-1.33
4.50	4012500.00	4003420.68	-0.22	3993825.95	-0.46	3951711.24	-1.51
4.75	4128125.00	4125895.38	-0.05	4093754.75	-0.83	4084808.11	-1.05
5.00	4250000.00	4257194.29	0.16	4192080.13	-1.36	4096803.88	-3.60

For N=4, a₁=2997608.684
a₂=33190.75084
a₃=29232.59062
a₄=2902.536744

For N=5, a₁=3035721.67
a₂=-16210.2870
a₃= 27733.4034
a₄=12343.6674
a₅=-1598.21379

$$F(t) = 932 + 40t \quad (3.38)$$

$$G(t) = 3 \times 10^6 + 5 \times 10^4 t^2 \quad (3.39)$$

In these tables, the computational errors are calculated using the following definition:

$$\text{Error} = 1 - \frac{p}{q} \quad (3.40)$$

where p and q represent retrieved and imposed true conditions, respectively.

As described in Table 3.3, those in Tables 3.8 and 3.11 are for the medium initially at phase change temperature, while those in Tables 3.9 and 3.10 are for the medium initially subcooled at 300 K. The former are thus one phase problems while the latter are two-phase problems. Again both series and incremental solution methods are used for solution and their results are good. For example in Table 3.8, the series solution results converge even with a value of N that is as low as 3, whereas in seeking the flux condition in Table 3.9, the series converges rapidly from $N=4$ to $N=5$ (higher degree results not shown). In Tables 3.8 and 3.11 the medium melts as soon as the boundary conditions are imposed, whereas in Tables 3.9 the medium starts to melt at time greater than 4 seconds. In all cases, the accuracy of the results appears to be unaffected by the time when melting takes place. The results in Tables 3.10 and 3.11 follow the same trends as the ones in Table 3.8 and 3.9. Tests for the time variant conditions are thus successful.

The inverse solution techniques developed in this study are expected to be accurate as mentioned earlier that is close to the end of the previous section. Yet for the Stefan problems solved in this work, the accuracy of the techniques is better at large time than small time. This can be attributed to the curvature of the interface position curve, which is always large at small time [2]; see Figure 3.2. Then, in the numerical solution of (3.27), there will be a slight error associated with the linearization of the position at small time. At large time, however, the position curve tends to be linear; the linearization error will be diminished. In fact, as time progresses, the accurate terms under the summation in (3.27) rapidly outnumber the inaccurate terms to the effect that the boundary conditions can always be evaluated accurately with the present method at large time; see the results in Tables 3.4 through 3.11. This is a distinct departure from the trends of other time marching schemes reported in the literature in which the errors tend to grow with time. It should also be pointed out that, for the Stefan-Neumann problem chosen for comparison in the present study, there is a singularity of the temperature at zero time. This also contributes to the large discrepancy of the results at small time, which must not be overlooked.

It has been firmly established that, in the solution of the Stefan problems, only the Stefan-Neumann problems can be solved exactly. Yet, it is also possible to develop an exact solution for an exponential condition imposed on the boundary; such condition, however, has been considered as physically untenable in the literature [7]. Worse yet, such a condition gives rise to a

constant velocity of the interface, a situation making the present linearization scheme exact in the solution of the inverse problems [5]. Perfect results will be obtained, rendering the test of the exponential conditions meaningless.

3.7 Extension and Concluding Remarks

The analysis developed in this chapter can be readily extended for the solution of inverse problems with multiple phases. For such problems, times for re-melt and re-freeze of the medium must be closely accounted for, and the analyses [4,65] can be readily adapted for the development of the inverse solution techniques presented in this study. The present analysis can also be extended for the solution of inverse problems in which the properties are unequal for different phases of the medium. For such problems, double source and sink fronts must be used as given in the solution of the regular problems in reference 17. Finally, it is noted that although problems in one-dimensional, semi-infinite domain have been solved for examples in this work, problems in finite domains (e.g., plane wall) can also be solved with the present methods. For these problems, there are two boundaries and two boundary conditions are imposed. Two unknowns are thus sought simultaneously, and this requires the input of one additional condition in the form of either temperature or heat flux at any interior point close to the boundary where no phase change takes place. On the other hand, for problems with two phase change interfaces caused by separate heat input simultaneously from both sides of the boundaries, the interface motion data for the second interface will serve as this additional

condition. In any event, no flux information is needed at both sides of the interfaces. Furthermore, the present method can also be applied to the solution of problems in multiple dimensions. Again the inverse solution for these problems can be developed on the basis of the solution of regular versions of these problems.

CHAPTER IV
SOLUTION OF ABLATION PROBLEMS WITH ONE MOVING BOUNDARY
BY A SOURCE-AND-SINK METHOD

It is the purpose of this chapter to present a source-and-sink method for the solution of ablation problems with one moving boundary. As will be shown, the essential feature of this method is that the solution will be sought in a fixed domain which does not change with time. In fact, the ablated region is to be treated as a fictitious domain in which the flux condition at the ablated boundary will match that of the imposed condition at the moving boundary. Then, with the additional vaporization temperature given for the moving boundary, the conditions at this boundary are overspecified for the fictitious domain. The problem can thus be taken as an inverse problem in the sense that the condition on the fixed boundary is sought during the ablation period. This condition will provide for the necessary conditions at the moving boundary for the solution of the problem.

4.1 Solution Methodology

For a subcooled medium, the problem can be divided into two stages; namely, pre-ablation stage and ablation stage. During the pre-ablation stage, heat is added to the surface of the medium in order to raise its temperature to the phase-change temperature. Then with continuous heating, ablation takes place. The ablated

surface moves inward with time and the imposed heat flux follows this boundary motion.

As shown in Figure 4.1, it is assumed that the medium is homogeneous and isotropic. The thermophysical properties are constant. For the ablation problem, the ablated region is immediately removed upon formation. Radiation is taken to be a surface phenomenon. It is also assumed that the medium changes phase at a distinct temperature; that is, no mushy zone in the medium. Moreover, there is no volumetric heat generation in the medium. Under these assumptions, the ablation problem can be formulated as follows:

Pre-ablation Stage

Governing equation--

$$\begin{aligned} \bar{r} &\in (A \cup S) \\ \nabla^2 T_0(\bar{r}, t) &= \frac{1}{\alpha} \frac{\partial T_0(\bar{r}, t)}{\partial t}, \\ t_0 &\geq t > 0 \end{aligned} \quad (4.1)$$

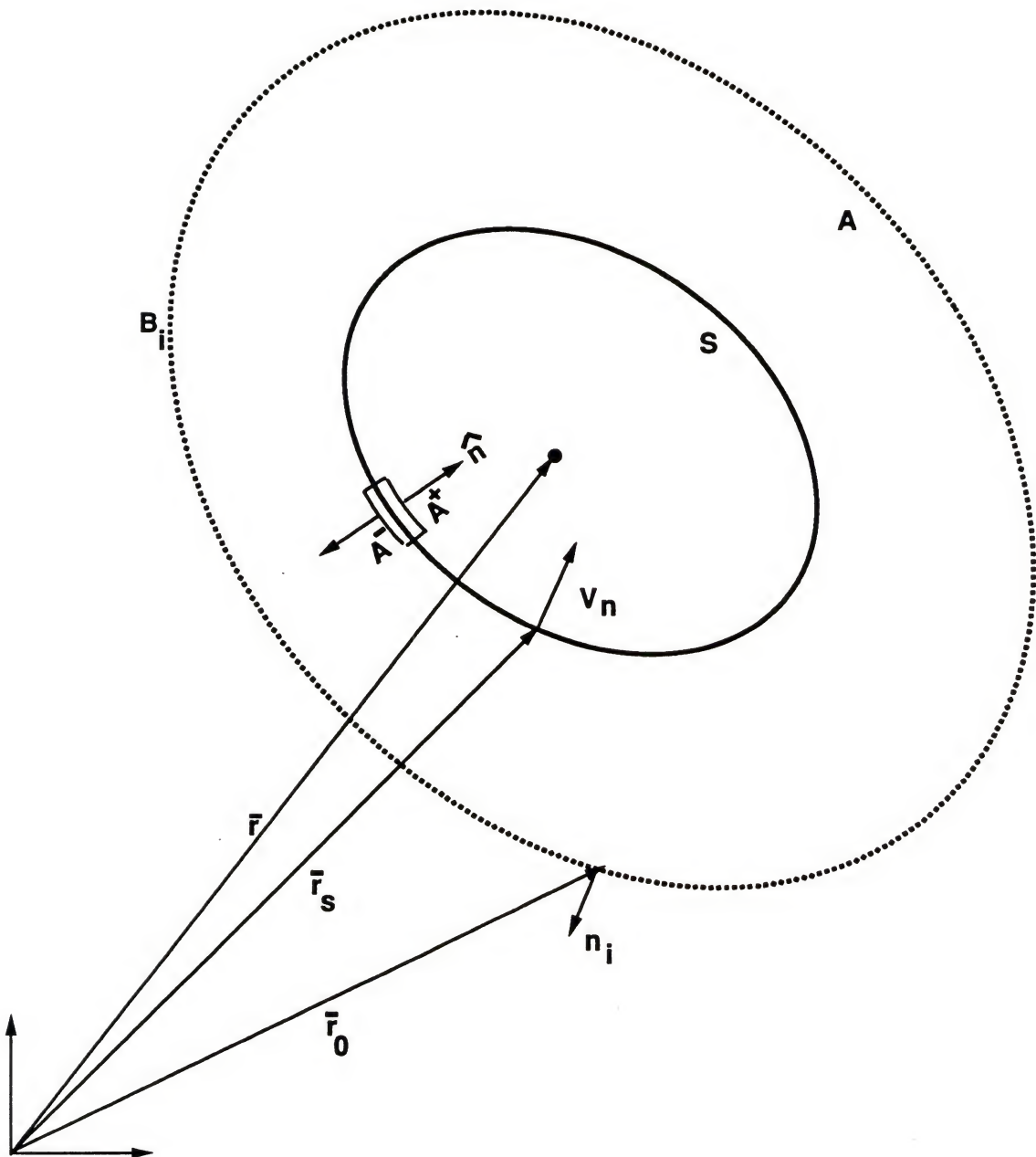
Initial condition--

$$T_0(\bar{r}, 0) = T_i(\bar{r}) \quad (4.2)$$

Boundary condition--

$$G(t) = -k \frac{\partial T_0(\bar{r}_0, t)}{\partial n_i}, \quad (4.3)$$

Figure 4.1 System analyzed



Ablation Stage

Solid Region:

Governing equation--

$$\begin{aligned} \bar{r} \in S \\ \nabla^2 T_S(\bar{r}, t) = \frac{1}{\alpha} \frac{\partial T_S(\bar{r}, t)}{\partial t}, \\ t > t_0 \end{aligned} \tag{4.4}$$

Initial condition--

$$T_S(\bar{r}, t_0) = T_0(\bar{r}, t_0) \tag{4.5}$$

Boundary conditions--

$$T_S(\bar{r}_S, t) = T_v \tag{4.6}$$

$$G(t) = -k \frac{\partial T_S(\bar{r}_S, t)}{\partial n} + \rho L \bar{v} \cdot \hat{n} \tag{4.7}$$

where all notations have their usual meaning; see nomenclature. Here \bar{r}_S represents the position for the ablated surface, which is unknown for the ablation problem; t_0 is the time when ablation starts; and $T_S(\bar{r}, t_0)$ is the temperature distribution in the medium at the onset of ablation. Both t_0 and $T_0(\bar{r}, t_0)$ are to be found from the solution of pre-ablation stage. Notice that the boundary flux condition, equation (4.7), which relates the ablation velocity to the temperature gradient, is the only source of nonlinearity in the ablation problem.

The ablation problem above is solved by using a source-and-sink method. In this method, the domain for investigation is extended to cover AUS; the ablation front becomes an interior phase-change interface. Solution is thus rendered in a fixed domain, in which the ablated region is fictitious in the sense that it is physically nonexistent; however, the unablated (solid) region is real. The conditions imposed at the moving boundary in the original ablation problem are taken to be those imposed on the interior phase-change interface. Thus, for the new problem in the fixed domain, the interface position and the fictitious condition on the fixed boundary are the unknowns to be determined. Once they are found, they can be used for input in the determination of the temperature in the solid region. The problem is thus solved totally. An equivalent problem is formulated with a source-and-sink method as follows:

Equivalent Problem

Governing equation--

$$\begin{aligned} \bar{r} &\in (A \cup S) \\ \nabla^2 T(\bar{r}, t) - \frac{\rho L}{k} \bar{v} \cdot \hat{n} \delta(\bar{r} - \bar{r}_S) &= \frac{1}{\alpha} \frac{\partial T(\bar{r}, t)}{\partial t}, \\ t &> t_0 \end{aligned} \tag{4.8}$$

Initial condition--

$$T(\bar{r}, t_0) = T_0(\bar{r}, t_0) \tag{4.9}$$

Interface conditions--

$$T(\bar{r}_S, t) = T_v \quad (4.10)$$

$$G(t) = -k \frac{\partial T(\bar{r}_S, t)}{\partial n} + \rho L \bar{v} \cdot \hat{n} \quad (4.11)$$

where $\delta(\bar{r} - \bar{r}_S)$ denotes a Dirac delta function. As has been shown in Chapter 3, equation (4.8) reduces to (4.4). Also integrating (4.8) across the interface at \bar{r}_S enables the $G(t)$ to represent the heat flux imposed on the interface at the fictitious side.

Equation (4.8) can be solved by using (3.18) in which the domains for integration for the first and second terms on the right are changed to $A \cup S$. Also minus sign is taken for the Dirac delta function term in which \bar{r}_f is changed to \bar{r}_S as

$$T(\bar{r}, t) = \int_{A \cup S} G(\bar{r}, t | \bar{r}', t_0) T_i(\bar{r}') dV' - \frac{L}{c} \int_{t_0}^t \int_{A \cup S} G(\bar{r}, t | \bar{r}', \tau) \bar{v} \cdot \hat{n} \delta(\bar{r}' - \bar{r}_S) dV' d\tau + \sum_i \left\{ \right\} \quad (4.12)$$

This temperature equation is forced to satisfy the interface temperature equation (4.10) by setting \bar{r} to the ablated surface position, \bar{r}_S , and $T(\bar{r}_S, t)$ to the phase-change temperature, T_v , as

$$T_v = \int_{A \cup S} G(\bar{r}_S, t | \bar{r}', t_0) T_i(\bar{r}') dV' - \frac{L}{c} \int_{t_0}^t \int_{A \cup S} G(\bar{r}_S, t | \bar{r}', \tau) \bar{v} \cdot \hat{n} \delta(\bar{r}' - \bar{r}_S) dV' d\tau + \sum_i \left\{ \right\} \quad (4.13)$$

Equation (4.12) is also differentiated to satisfy (4.11) as

$$\begin{aligned}
G(t) = & -k \left\{ \int_{A \cup S} \frac{\partial G(\bar{r}_S, t | \bar{r}', t_0)}{\partial n} T_i(\bar{r}') dV' - \frac{L}{c} \int_{t_0}^t \int_{A \cup S} \frac{\partial G(\bar{r}_S, t | \bar{r}', \tau)}{\partial n} \bar{v} \cdot \hat{n} \delta(\bar{r}' - \bar{r}_S) dV' d\tau \right. \\
& \left. + \sum_i \frac{\partial}{\partial n} \left\{ \right\}_{\bar{r}=\bar{r}_S} \right\} + \rho L \bar{v} \cdot \hat{n}
\end{aligned} \tag{4.14}$$

where differentiation is to be effected at the solid side of the interface. Notice that the boundary condition is embodied in the summation term, which, according to Table 3.1, should be expressed in the form

$$\left\{ \right\} = \alpha \int_0^t \int_{\bar{r}_0} G(\bar{r}, t | \bar{r}', \tau) \frac{\partial T(\bar{r}', t)}{\partial n_i} d\bar{r}' d\tau \tag{4.15}$$

where the following condition is taken at the fixed boundary:

$$\frac{\partial T(\bar{r}', t)}{\partial n_i} = \frac{-G(\bar{r}', t)}{k} \quad \text{for } t \leq t_0 \quad \text{and} \quad \frac{-g(\bar{r}', t)}{k} \quad \text{for } t > t_0 \tag{4.16}$$

where $g(\bar{r}', t)$ represents the imposed fictitious heat flux. Clearly, there are two unknowns in (4.13) and (4.14): $g(\bar{r}', t)$ and \bar{v} ; there are two equations to solve them. In this effort, t_0 and $T_0(\bar{r}, t_0)$ are to be found from the solution of a pre-ablation problem as described previously.

4.2 Illustrative Examples

Use is now made of the solution methodology given in the preceding section to solve examples in semi-infinite medium. Four examples will be provided and they include medium with or without subcooling with the boundary imposed with constant or variable heat flux conditions.

For a subcooled medium, the pre-ablation stage solution has been given in Chapter 3. The pre-ablation temperature can be taken from (3.22) as

$$T_0(x,t) = T_i + \frac{1}{k} \sqrt{\frac{\alpha}{\pi}} \int_0^t \frac{G(s)}{(t-s)^{1/2}} \exp\left[-\frac{x^2}{4\alpha(t-s)}\right] ds \quad (4.17)$$

Thus the time when ablation starts (t_0) can be found by solving the following equation implicitly

$$T_v = T_i + \frac{1}{k} \sqrt{\frac{\alpha}{\pi}} \int_0^{t_0} \frac{G(s)}{(t_0-s)^{1/2}} ds \quad (4.18)$$

Also at the moment when ablation starts, the temperature in the solid region is given by the relation

$$T_0(x,t_0) = T_i + \frac{1}{k} \sqrt{\frac{\alpha}{\pi}} \int_0^{t_0} \frac{G(s)}{(t_0-s)^{1/2}} \exp\left[-\frac{x^2}{4\alpha(t_0-s)}\right] ds \quad (4.19)$$

which serves as the initial condition for the ablation stage.

For the problem at hand, (4.12) can be used to derive temperature as

$$\begin{aligned} T(x,t) = & T_i + \frac{1}{k} \sqrt{\frac{\alpha}{\pi}} \int_{\tau=0}^{t_0} \frac{G(\tau)}{(t-\tau)^{1/2}} \exp\left[-\frac{x^2}{4\alpha(t-\tau)}\right] d\tau + \frac{1}{k} \sqrt{\frac{\alpha}{\pi}} \int_{\tau=t_0}^t \frac{g(\tau)}{(t-\tau)^{1/2}} \exp\left[-\frac{x^2}{4\alpha(t-\tau)}\right] d\tau \\ & - \frac{L}{c} \int_{\tau=t_0}^t \frac{dR_1(\tau)}{d\tau} \frac{1}{\sqrt{4\pi\alpha(t-\tau)}} \left\{ \exp\left(-\frac{(x-R_1(\tau))^2}{4\alpha(t-\tau)}\right) + \exp\left(-\frac{(x+R_1(\tau))^2}{4\alpha(t-\tau)}\right) \right\} d\tau \end{aligned} \quad (4.20)$$

Also (4.13) is used to derive

$$\begin{aligned}
T_v = T_i + \frac{1}{k\sqrt{\pi}} \int_{\tau=0}^{t_0} \frac{G(\tau)}{(t-\tau)^{1/2}} \exp\left[-\frac{R_1^2(t)}{4\alpha(t-\tau)}\right] d\tau + \frac{1}{k\sqrt{\pi}} \int_{\tau=t_0}^t \frac{g(\tau)}{(t-\tau)^{1/2}} \exp\left[-\frac{R_1^2(t)}{4\alpha(t-\tau)}\right] d\tau \\
- \frac{L}{c} \int_{\tau=t_0}^t \frac{dR_1(\tau)}{d\tau} \frac{1}{\sqrt{4\pi\alpha(t-\tau)}} \left\{ \exp\left(-\frac{(R_1(t)-R_1(\tau))^2}{4\alpha(t-\tau)}\right) + \exp\left(-\frac{(R_1(t)+R_1(\tau))^2}{4\alpha(t-\tau)}\right) \right\} d\tau
\end{aligned} \tag{4.21}$$

Equation (4.14) follows as

$$\begin{aligned}
G(t) = \frac{R_1(t)}{2\sqrt{\pi\alpha}} \int_{\tau=0}^{t_0} \frac{G(\tau)}{(t-\tau)^{3/2}} \exp\left[-\frac{R_1^2(t)}{4\alpha(t-\tau)}\right] d\tau + \frac{R_1(t)}{2\sqrt{\pi\alpha}} \int_{\tau=t_0}^t \frac{g(\tau)}{(t-\tau)^{3/2}} \exp\left[-\frac{R_1^2(t)}{4\alpha(t-\tau)}\right] d\tau \\
- \frac{L\rho}{4\sqrt{\pi\alpha}} \int_{\tau=t_0}^t \frac{dR_1(\tau)}{d\tau} \frac{1}{(t-\tau)^{3/2}} \left\{ (R_1(t)-R_1(\tau)) \exp\left(-\frac{(R_1(t)-R_1(\tau))^2}{4\alpha(t-\tau)}\right) \right. \\
\left. + (R_1(t)+R_1(\tau)) \exp\left(-\frac{(R_1(t)+R_1(\tau))^2}{4\alpha(t-\tau)}\right) \right\} d\tau + \rho L \frac{dR_1(t)}{dt}
\end{aligned} \tag{4.22}$$

Solution of (4.21) and (4.22) subject to the initial condition, $R_1(t_0)=0$, yields the results for two unknown functions $R_1(t)$ and $g(t)$.

4.3 Numerical Solution of Ablated Front

Equations (4.21) and (4.22) are coupled nonlinear integrodifferential equations, which will be solved numerically. In this effort, a local linearization scheme is employed as described in Section 3.4. For the present problem, the entire time range is divided into small increments in which both $\frac{dR_1(t)}{dt}$ and $g(t)$ are treated as constants. They can thus be taken out of their respective integrals, and the convolution integrals in these equations are changed to summations as

$$\begin{aligned}
T_v = T_i &+ \frac{1}{k} \sqrt{\frac{\alpha}{\pi}} \int_{\tau=0}^{t_0} \frac{G(\tau)}{(t_N - \tau)^{1/2}} \exp\left[-\frac{R_1^2(t_N)}{4\alpha(t_N - \tau)}\right] d\tau \\
&+ \frac{1}{k} \sqrt{\frac{\alpha}{\pi}} \sum_{n=1}^N g(t_n) \int_{\tau=t_{n-1}}^{t_n} \frac{1}{(t_N - \tau)^{1/2}} \exp\left[-\frac{R_1^2(t_N)}{4\alpha(t_N - \tau)}\right] d\tau \\
&- \frac{L}{c} \sum_{n=1}^N \frac{dR_1(t_n)}{dt} \int_{\tau=t_{n-1}}^{t_n} \frac{1}{\sqrt{4\pi\alpha(t_N - \tau)}} \left\{ \exp\left(-\frac{(R_1(t_N) - R_1(\tau))^2}{4\alpha(t_N - \tau)}\right) + \exp\left(-\frac{(R_1(t_N) + R_1(\tau))^2}{4\alpha(t_N - \tau)}\right) \right\} d\tau
\end{aligned} \tag{4.23}$$

and

$$\begin{aligned}
G(t_N) = \frac{R_1(t_N)}{2\sqrt{\pi\alpha}} &\left\{ \int_{\tau=0}^{t_0} \frac{G(\tau)}{(t_N - \tau)^{3/2}} \exp\left[-\frac{R_1^2(t_N)}{4\alpha(t_N - \tau)}\right] d\tau \right. \\
&+ \sum_{n=1}^N g(t_n) \int_{\tau=t_{n-1}}^{t_n} \frac{1}{(t_N - \tau)^{3/2}} \exp\left[-\frac{R_1^2(t_N)}{4\alpha(t_N - \tau)}\right] d\tau \Big\} \\
&- \frac{L\rho}{4\sqrt{\pi\alpha}} \sum_{n=1}^N \frac{dR_1(t_n)}{dt} \int_{\tau=t_{n-1}}^{t_n} \frac{1}{(t_N - \tau)^{3/2}} \left\{ (R_1(t_N) - R_1(\tau)) \exp\left(-\frac{(R_1(t_N) - R_1(\tau))^2}{4\alpha(t_N - \tau)}\right) \right. \\
&\left. + (R_1(t_N) + R_1(\tau)) \exp\left(-\frac{(R_1(t_N) + R_1(\tau))^2}{4\alpha(t_N - \tau)}\right) \right\} d\tau + \rho L \frac{dR_1(t_N)}{dt}
\end{aligned} \tag{4.24}$$

Notice that the removal of $\frac{dR_1(t)}{dt}$ and $g(t)$ will cause a slight error in the numerical solution of the ablated front position. However, R_1 is a gradual function of t , and so is $g(t)$ during the early stage of ablation. The error associated with the linearization is expected to be small. In practice, such an error can always be reduced by taking small time increments.

It is also noted that the ablated position $R_1(t)$ in the Green's function can be related to $R_1(t_{n-1})$ by means of the interface velocity, $\frac{dR_1(t_{n-1})}{dt}$, as shown in Figure 3.2. The position R_1 at any time t should thus be discounted as unknown. Numerically, the $R_1(t)$ and $g(t)$ in these equations will be solved incrementally starting from t_0 until the final time t_N is reached. This corresponds to a time marching scheme in the numerical solution. Computer programs developed for the solution of the ablation problems given in this chapter are compiled in Appendix F.

4.4 Results and Discussion

Four examples are tested and they include semi-infinite medium with and without subcooling with the moving boundary imposed with constant and time-variant conditions (see Table 4.1 for summary). Notice that, in this table, under the column of problem description, the ablated region is always counted as one phase. Thus, for a medium initially at phase-change temperature imposed with a constant heat-flux condition, such as Example 1, there is one ablated phase. It is thus called one-phase problem in the table.

Example 1 deals with an ablation problem that can be solved exactly. As shown in Carslaw and Jaeger [7], for a semi-infinite medium ablated with a constant velocity U , the heat flux imposed on the boundary is given by the relation

$$G = [L + c(T_v - T_i)]\rho U \quad (4.25)$$

where T_i is constant and the velocity is related to the ablation position as

Table 4.1 Conditions tested in four examples

Problem Description	Material	Heat Flux Condition Imposed $G(t)$ (W/m^2); t (S)	Remarks
1. One-phase $T_i=932 \text{ K}=T_m=T_v$	Aluminum	$G(t)=5 \times 10^5$	Exact solution is available for this case.
2. Two-phase $T_i=882 \text{ K}<T_m=T_v$	Aluminum	$G(t)=2 \times 10^6$	No exact solution for this case; results compared with three analytical solutions in the literature.
3. Two-phase $T_i=882 \text{ K}<T_m=T_v$	Aluminum	$G(t)=8 \times 10^4 t$	No exact solution for this case; results compared with moment method in the literature.
4. Two-phase $T_i=882 \text{ K}<T_m=T_v$	Aluminum	$G(t)=2 \times 10^3 t^2$	No exact solution for this case; results compared with moment method in the literature.

$$U = \frac{R_1(t)}{t} \quad (4.26)$$

It is expected that, for a medium initially at the phase-change temperature, G is equal to ρLU product. It is also noted that equation (4.25) is strictly valid for the medium in a quasi-steady state. At that time, the temperature in the solid is stabilized given as

$$T(x,t) = T_v e^{-\frac{U}{\alpha}(x-Ut)} \quad (4.27)$$

which occurs for the medium having ablated for a long period of time. The first two examples are thus chosen to substantiate these points.

For the first example, the ablation is due to melting. The medium is initially at the melting temperature so that the surface ablates as soon as the heat is applied. The position of the ablated surface is shown in Figure 4.2, where the curve appears to be linear. Using the exact solution (4.25) for comparison yields the error curves shown in Figure 4.3. Clearly, the results are stable and converge satisfactorily, errors being less than one tenth of one percent for all the time increments tested. Also the errors are slightly larger at small time, a result of the singularity associated with the heat flux that is abruptly applied on the surface at time zero. In what follows in the numerical computation of all the examples to be presented in this chapter, the time increment is taken as 0.5 sec, unless otherwise noted.

Example 1 provides an excellent test for the algorithms

Figure 4.2

Trend of ablated surface position for a medium initially at phase-change temperature imposed with a constant heat-flux condition

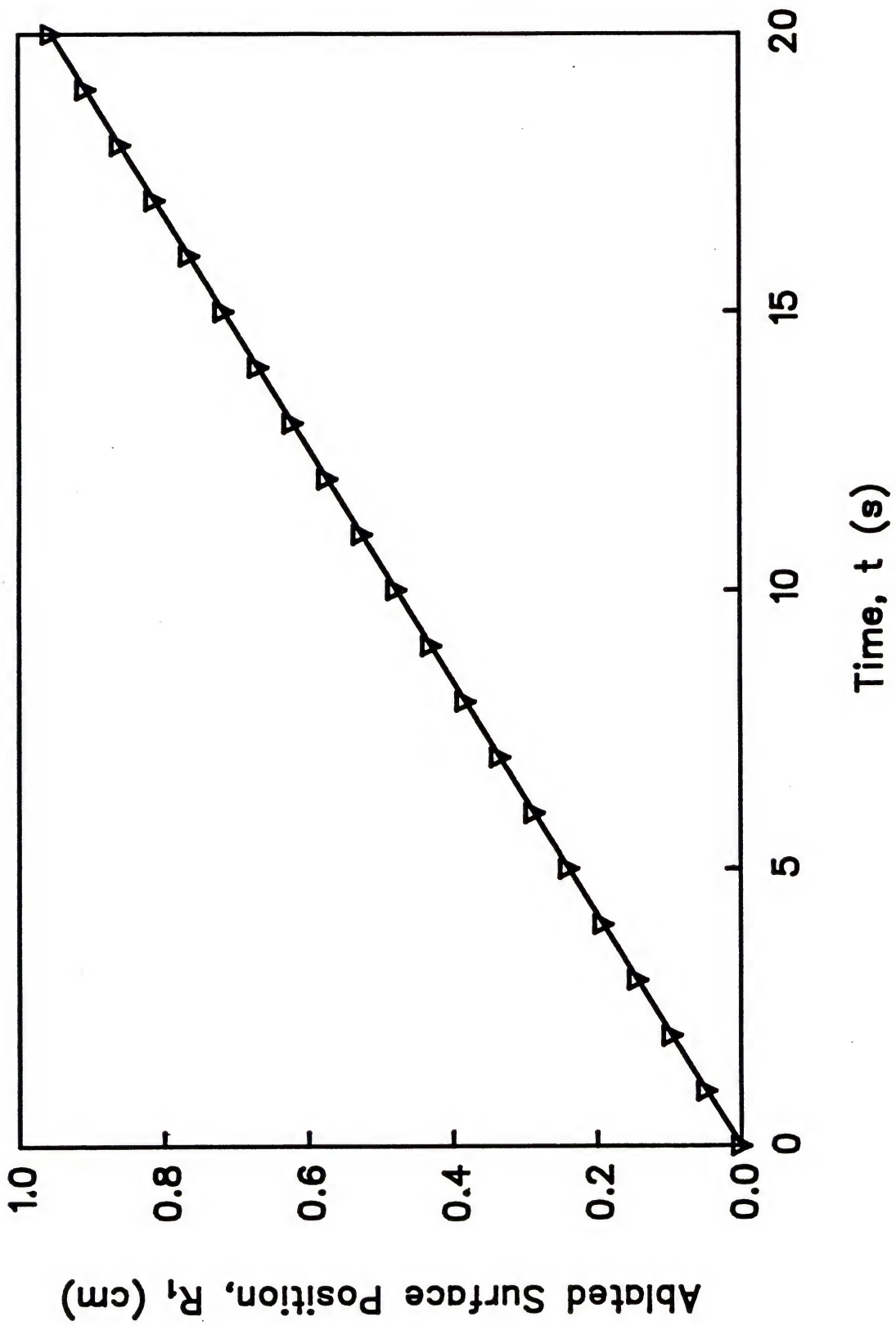
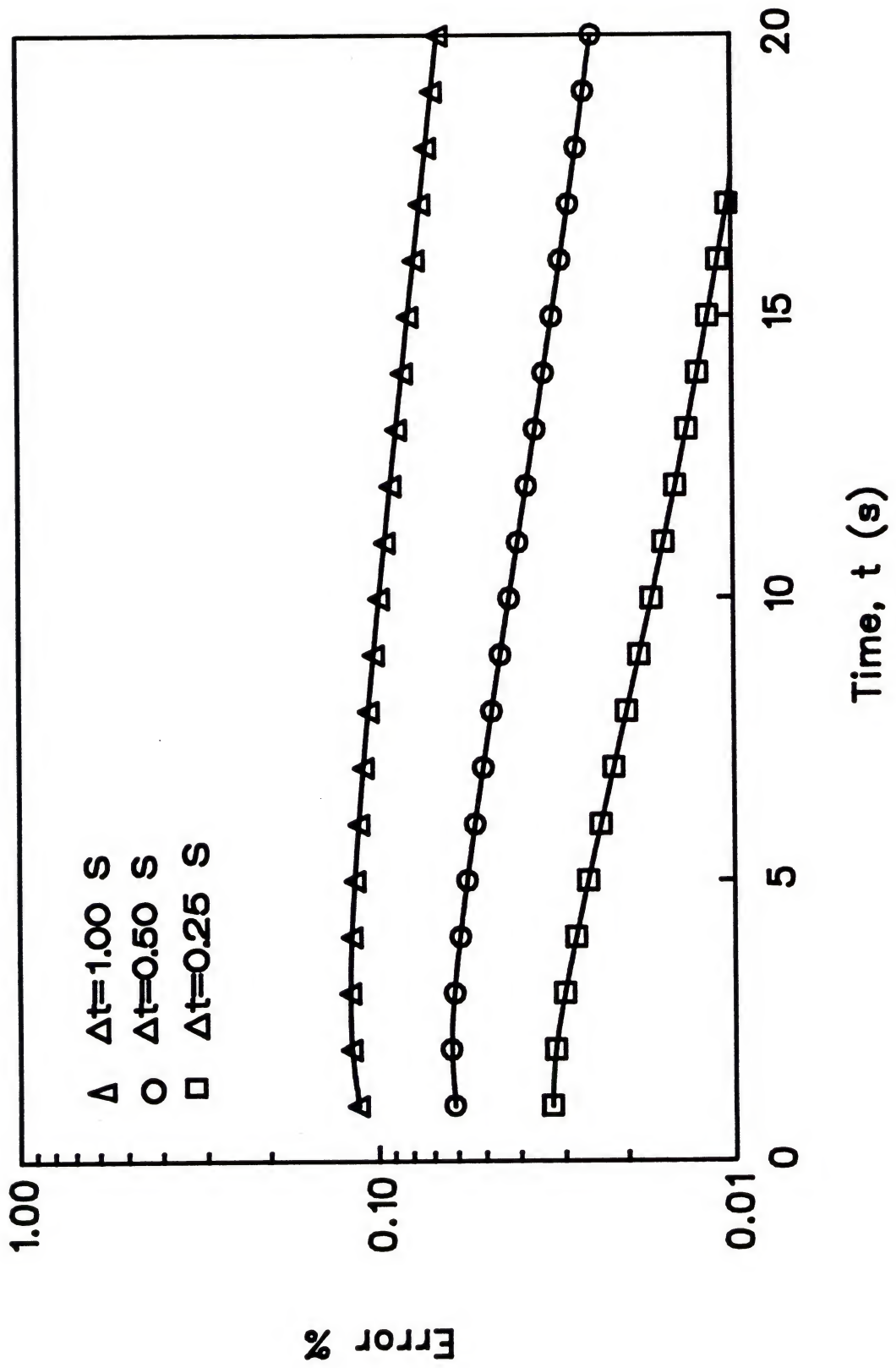


Figure 4.3 Accuracy, stability and convergence test of the SSM in the solution of ablation for a medium initially at phase-change temperature imposed with a constant heat-flux condition



developed for the solution of the ablation problems in this dissertation. In fact, the good results obtained in the first example are somewhat expected because of the linearization used in the solution of the integrodifferential equations for which the ablation rate itself is linear.

Example 2 is different because it deals with a medium which is initially subcooled (see Table 4.1). This time, there is no exact solution for checking the results, which have also been tested for convergence in Figure 4.4. Of interest in this figure is the slightly smaller ablation rate at small time, which can be attributed to the large slope of the temperature curve at the solid side of the moving boundary (not shown in dissertation). At large time, this slope diminishes, and finally it is stabilized as evidenced by the linearity of the position curve at large time. Using the numerical data over the last time step in Figure 4.4 yields an ablation rate of 0.00161 m/s, which is in good agreement with that computed using equation (4.25), error being less than 2.5 %. This gives a good indication of the ablation approaching the quasi-steady state.

An effort is also made to check the results of Example 2 with the analytical solutions reported in the literature. As shown in Figure 4.5, both the heat integral method (HIM) of Vallerani [40] and the moment method (MOM) of Zien [41] yield results that are in close agreement with the present source-and-sink method. Landau's results also fall right on the curve and have thus been omitted in the plot. Example 2 has thus been tested successfully.

Examples 3 and 4 deal with a subcooled medium imposed with

Figure 4.4

Stability and convergence test of the SSM
in the solution of ablation for a subcooled
medium imposed with a constant heat-flux
condition

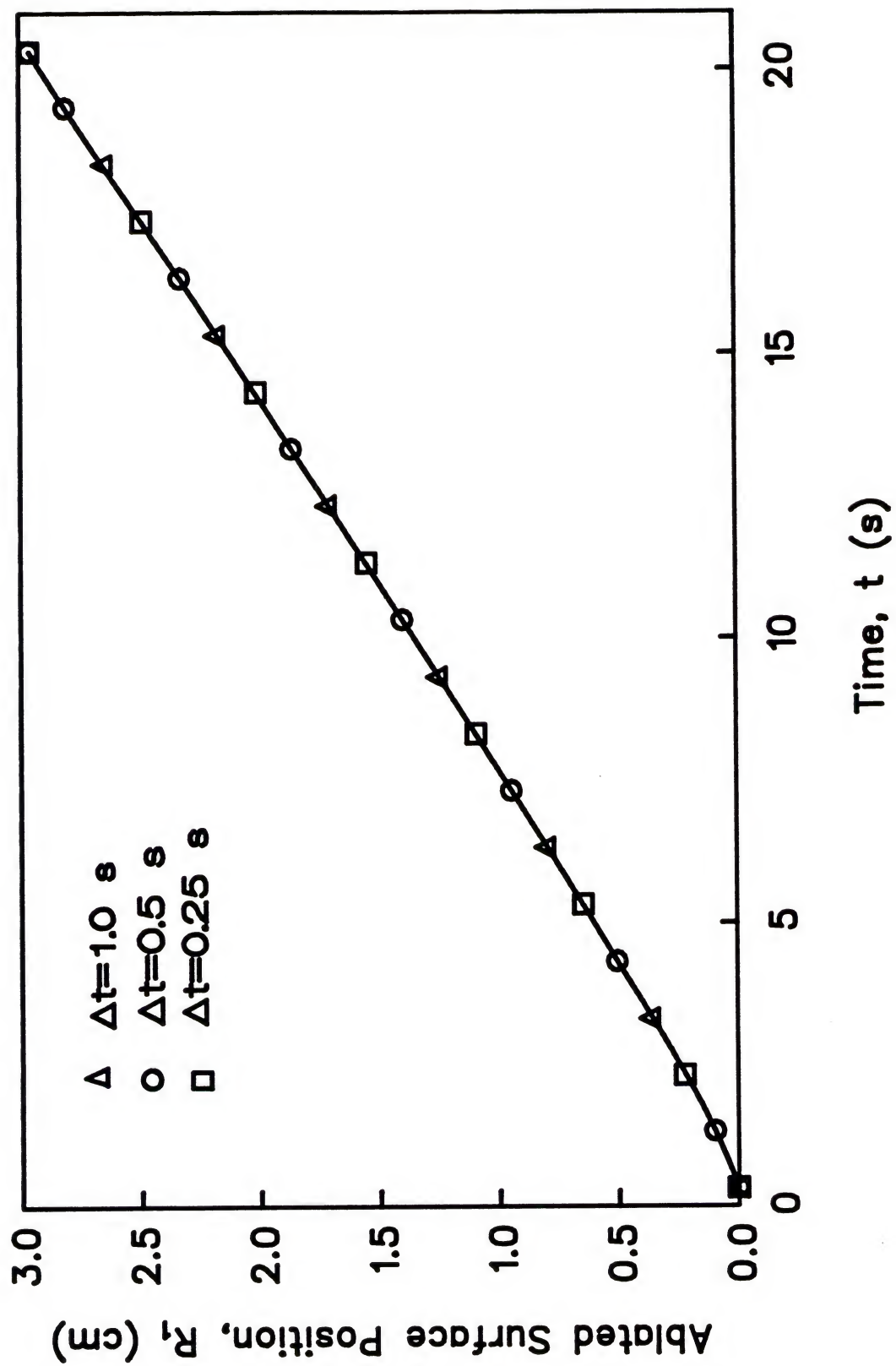
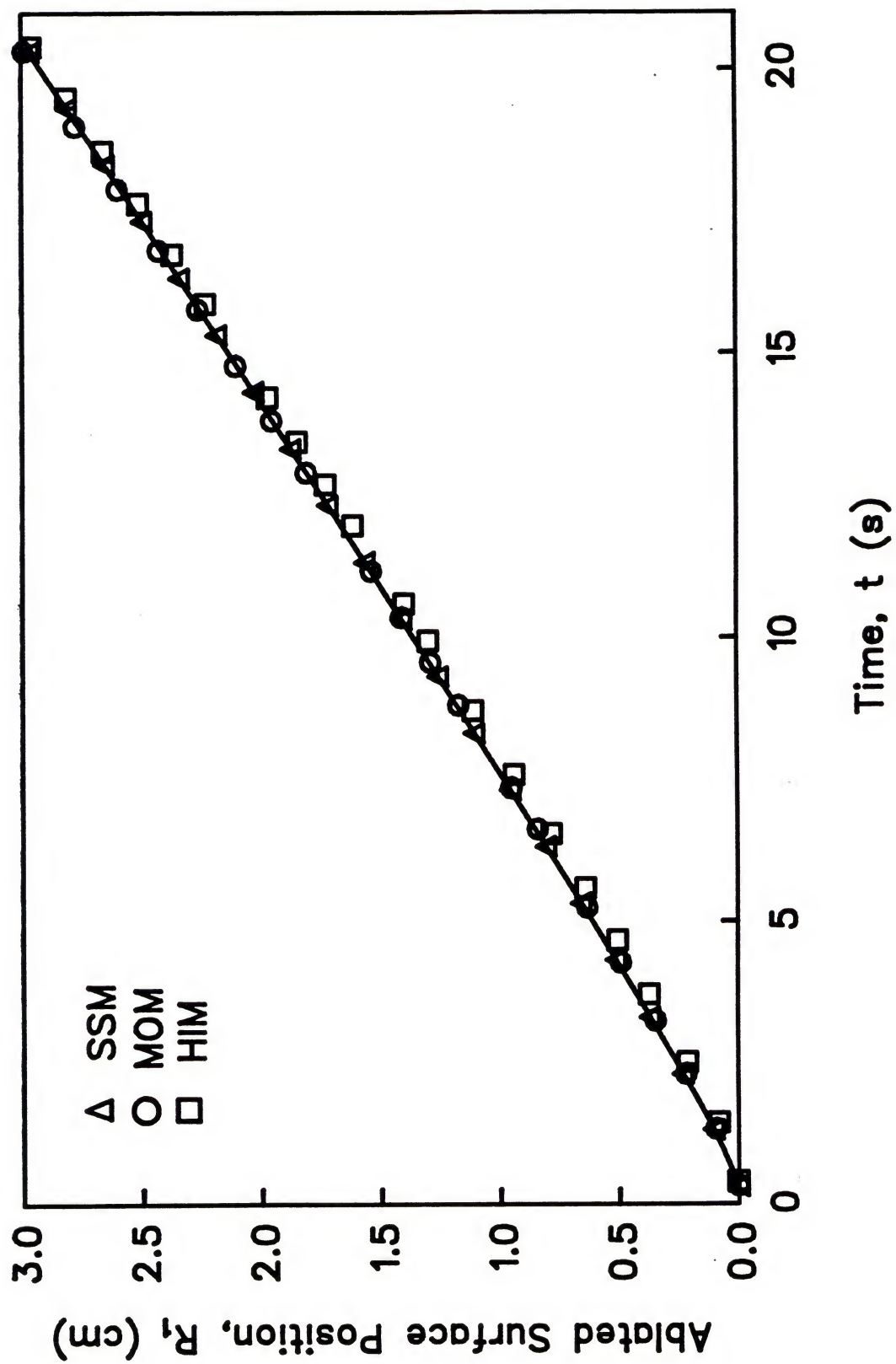


Figure 4.5

Comparison of ablated surface position with
two methods in the literature for a subcooled
medium imposed with a constant heat-flux
condition



time-variant heat-flux conditions. A linear heat-flux is imposed in Example 3, while a quadratic heat-flux is imposed in Example 4. Like Example 2, the medium is subcooled initially; there are two phases in the medium for both examples. Comparison of the present solution with the moment method for the linear heat-flux condition is given in Figure 4.6; convergence test based on the change of time increments for this condition is given in Figure 4.7. As shown in both figures, the results for the source-and-sink method are in close agreement with the moment method, and the results have converged with a time step as large as 0.5 sec. Similar results are seen in Figures 4.8 and 4.9 for the case of quadratic heat-flux. The tests for ablation problems with one moving boundary have thus been completed satisfactorily.

Figure 4.6

Comparison of ablated surface position with
the moment method in the literature for a
subcooled medium imposed with a linear heat
-flux condition

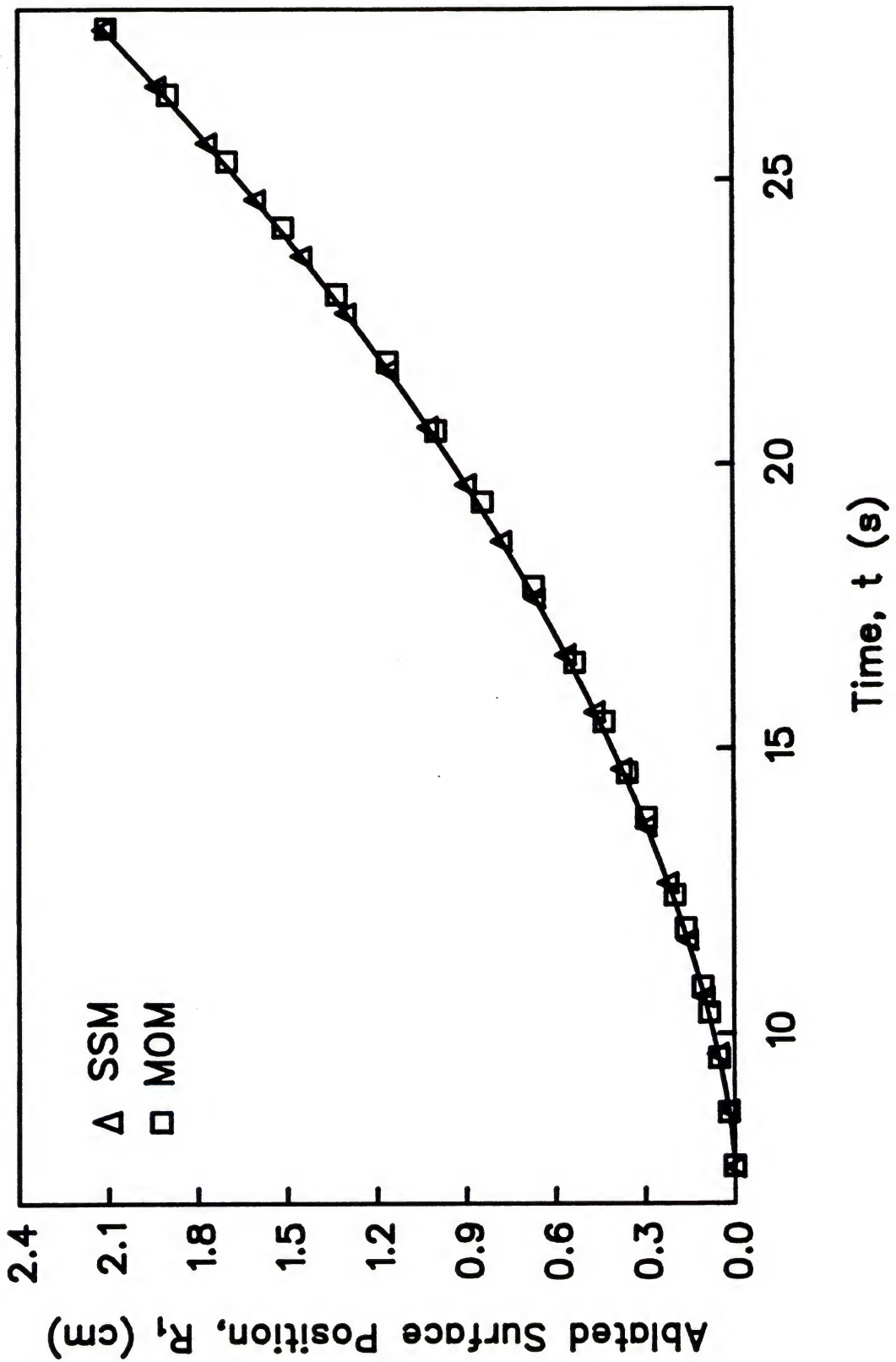


Figure 4.7

Stability and convergence test of the SSM
in the solution of ablation for a subcooled
medium imposed with a linear heat-flux
condition

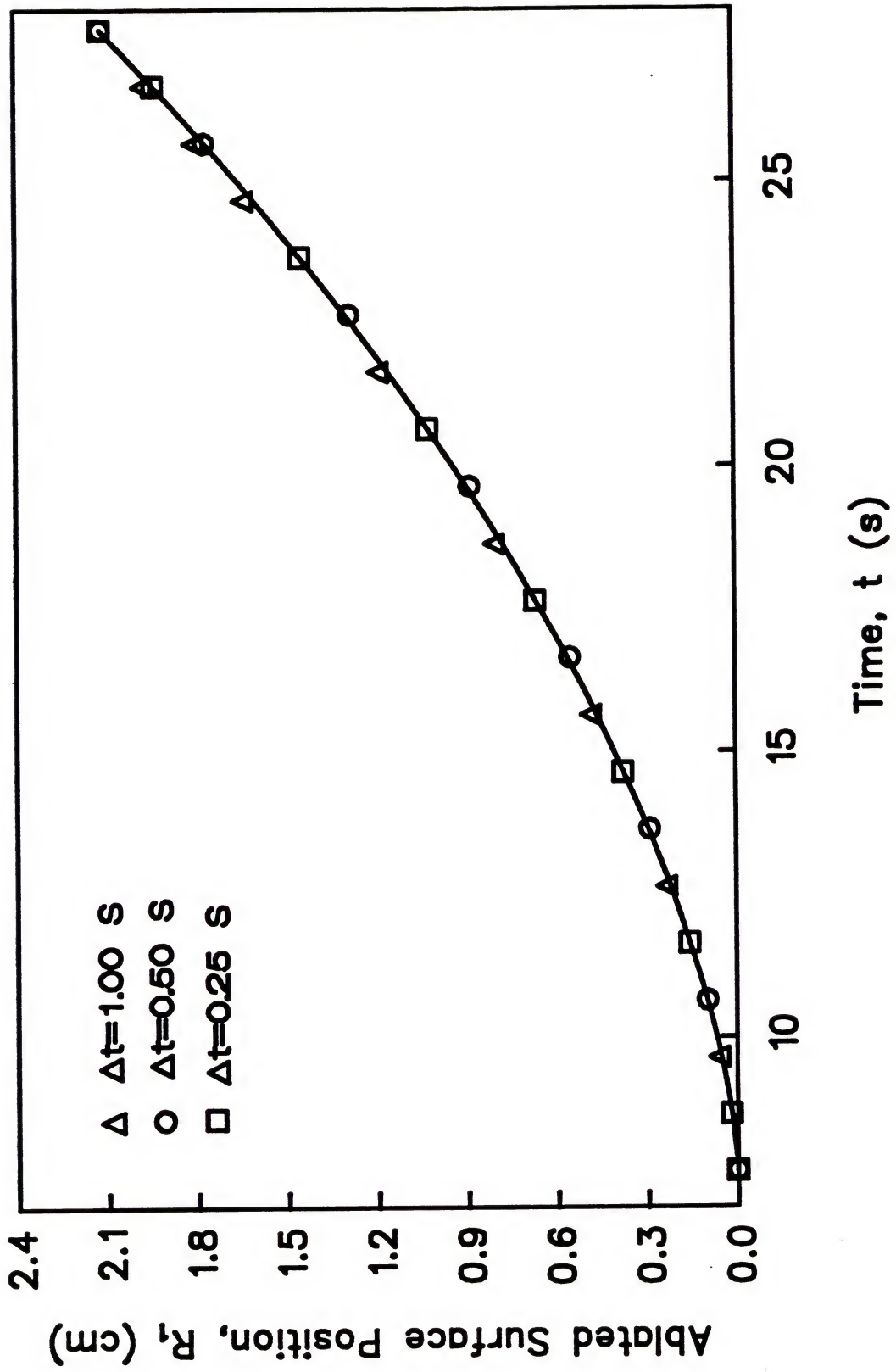


Figure 4.8 Comparison of ablated surface position with the moment method in the literature for a subcooled medium imposed with a quadratic heat-flux condition

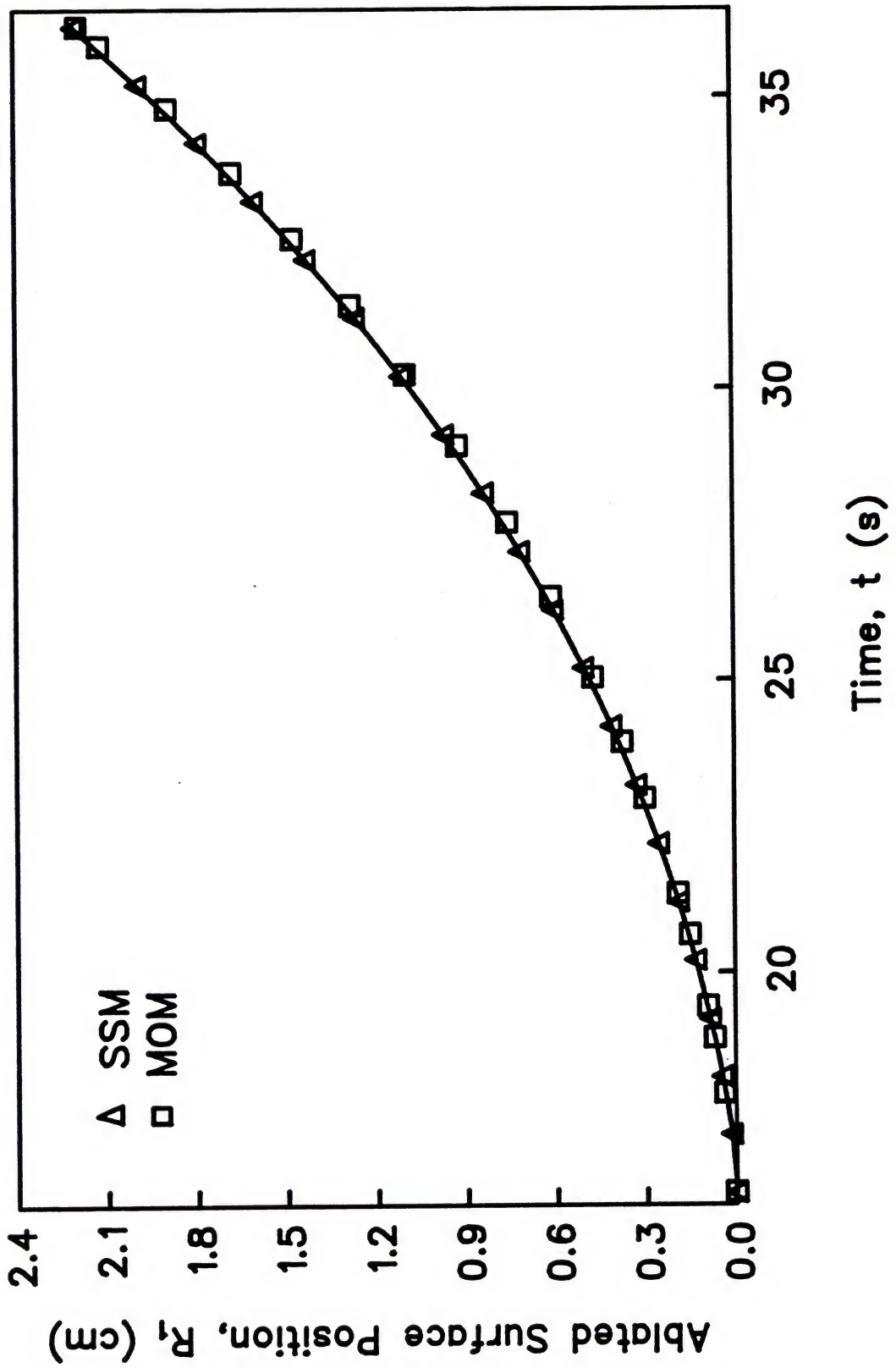
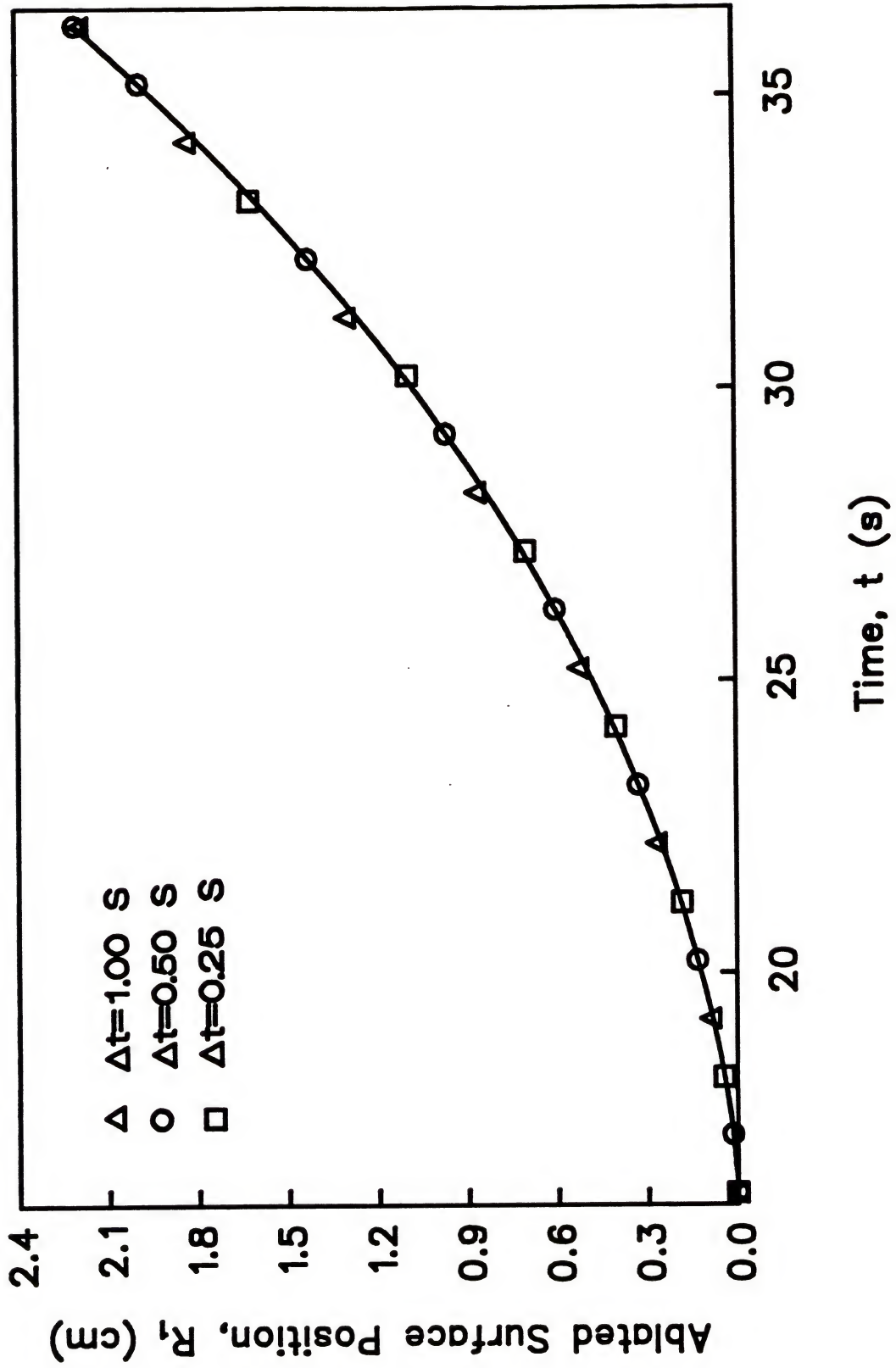


Figure 4.9

Stability and convergence test of the SSM
in the solution of ablation for a subcooled
medium imposed with a quadratic heat-flux
condition



CHAPTER V
SOLUTION OF ABLATION PROBLEMS WITH TWO MOVING BOUNDARIES
BY A SOURCE-AND-SINK METHOD

In this chapter, the source-and-sink method is used to solve more general ablation problems in which ablation occurs at the surface, while, at the same time, the solid melts and the melting front moves inward with time. In these problems, there will be three phases occurring simultaneously: the ablation layer at the surface, the liquid layer near the surface, and the solid region deep inside the body. There are two moving boundaries: one is the moving ablated surface and the other is the moving solid-liquid interface. Their positions are unknown *a priori* and must be determined as a part of the solution. These problems can be considered as the combination problems since they incorporate all the features of the ablation and Stefan problems studied earlier.

Using the same concept employed in solving the ablation problems with one moving boundary in Chapter IV, the ablated region is again considered as a fictitious region. The heat flux applied on the ablated boundary is then used to match the imposed condition at the moving boundary. This condition together with the vaporization and melting temperatures at the ablated and melting fronts, respectively, give sufficient information for the solution of the problem.

5.1 General Analysis

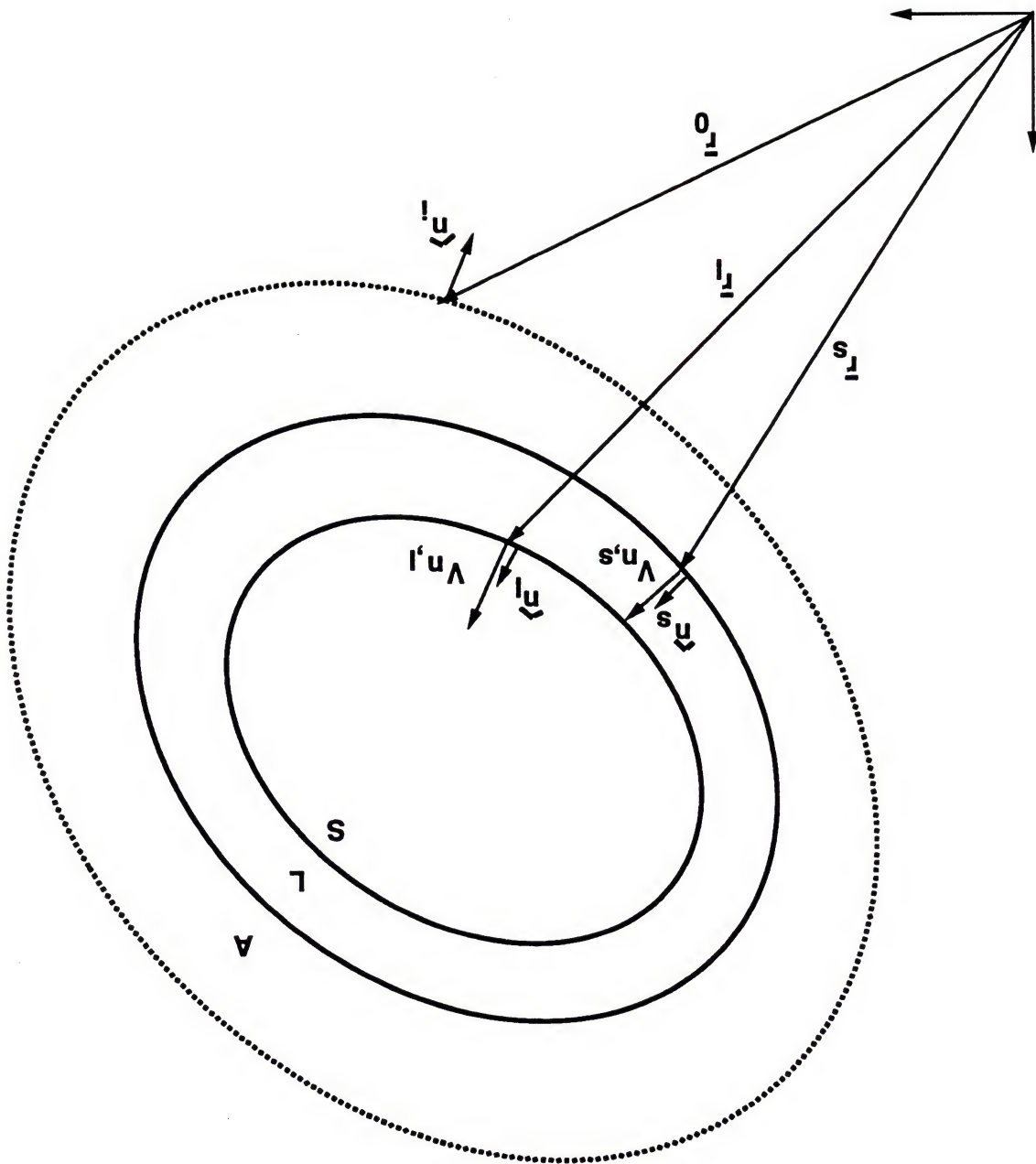
Considering a subcooled medium, the present ablation problem consists of three stages: pre-melt stage, melting stage, and ablation stage. In the pre-melt stage, heat is added to the surface of the medium to raise its temperature to the phase-change temperature. Then as heat continues, melting stage commences with the melting front starts at the surface and moves inward with time. With further addition of heat at a sufficiently high rate, vaporization may take place. It is assumed that the vapor is removed instantaneously upon formation (by being blown away, for instance) while the melting material stays in place. This is taken as the ablation stage.

Use is made of the system shown in Figure 5.1 for analysis. It is assumed that the thermophysical properties of the liquid and solid states are constant and of equal value. The medium is homogeneous and isotropic; phase changes take place at distinct temperatures. Then, in the absence of radiation, convection, and volumetric heat generation, the formulation of the problem can be given as follows.

Pre-melt Stage

Equations for this stage are the same as those for the same stage given earlier in Chapter IV; see equations (4.1-4.3). Note that, for the problem at hand, the domain in equation (4.1) should be changed to $\bar{r} \in (AULUS)$.

Figure 5.1 System analyzed



Melting Stage

Liquid Region:

Governing equation--

$$\begin{aligned} \bar{\mathbf{r}} &\in L \\ \nabla^2 T_L(\bar{\mathbf{r}}, t) &= \frac{1}{\alpha} \frac{\partial T_L(\bar{\mathbf{r}}, t)}{\partial t}, \\ t_0 &< t \leq t_v \end{aligned} \tag{5.1}$$

Boundary conditions--

$$G(t) = -k \frac{\partial T_L(\bar{\mathbf{r}}_0, t)}{\partial n_i} \tag{5.2}$$

Solid Region:

Governing equation--

$$\begin{aligned} \bar{\mathbf{r}} &\in S \\ \nabla^2 T_S(\bar{\mathbf{r}}, t) &= \frac{1}{\alpha} \frac{\partial T_S(\bar{\mathbf{r}}, t)}{\partial t}, \\ t_0 &< t \leq t_v \end{aligned} \tag{5.3}$$

Initial Conditions--

$$T_S(\bar{\mathbf{r}}, t_0) = T_0(\bar{\mathbf{r}}, t_0) \tag{5.4}$$

Interface conditions--

$$T_L(\bar{\mathbf{r}}_l, t) = T_S(\bar{\mathbf{r}}_l, t) = T_m \quad (5.5)$$

$$\frac{\partial T_S(\bar{\mathbf{r}}_l, t)}{\partial n_l} - \frac{\partial T_L(\bar{\mathbf{r}}_l, t)}{\partial n_l} = \frac{\rho L_f}{k} \bar{\mathbf{v}}_l \cdot \hat{\mathbf{n}}_l \quad (5.6)$$

$$\bar{\mathbf{r}}_l(t_0) = \bar{\mathbf{r}}_0 \quad (5.7)$$

Ablation Stage

Liquid Region:

Governing equation--

$$\begin{aligned} \bar{\mathbf{r}} &\in L \\ \nabla^2 T_{LL}(\bar{\mathbf{r}}, t) &= \frac{1}{\alpha} \frac{\partial T_{LL}(\bar{\mathbf{r}}, t)}{\partial t}, \\ t_v &< t \end{aligned} \quad (5.8)$$

Boundary Condition--

$$G(t) = -k \frac{\partial T_{LL}(\bar{\mathbf{r}}_S, t)}{\partial n_S} + \rho L_v \bar{\mathbf{v}}_S \cdot \hat{\mathbf{n}}_S \quad (5.9)$$

$$T_{LL}(\bar{\mathbf{r}}_S, t) = T_v \quad (5.10)$$

Initial condition--

$$T_{LL}(\bar{\mathbf{r}}, t_v) = T_L(\bar{\mathbf{r}}, t_v) \quad (5.11)$$

$$\bar{r}_S(t_v) = \bar{r}_0 \quad (5.12)$$

Solid Region:

Governing equation--

$$\begin{aligned} \bar{r} \in S \\ \nabla^2 T_{SS}(\bar{r}, t) = \frac{1}{\alpha} \frac{\partial T_{SS}(\bar{r}, t)}{\partial t}, \\ t_v < t \end{aligned} \quad (5.13)$$

Initial condition--

$$T_{SS}(\bar{r}, t_v) = T_S(\bar{r}, t_v) \quad (5.14)$$

Interface conditions--

$$T_{LL}(\bar{r}_l, t) = T_{SS}(\bar{r}_l, t) = T_m \quad (5.15)$$

$$\frac{\partial T_{SS}(\bar{r}_l, t)}{\partial n_l} - \frac{\partial T_{LL}(\bar{r}_l, t)}{\partial n_l} = \frac{\rho L_f}{k} \bar{v}_l \cdot \hat{n}_l \quad (5.16)$$

$$\bar{r}_l(t_v) = \bar{r}_v \quad (5.17)$$

where \bar{r}_0 denotes the original surface position, and \bar{r}_l and \bar{r}_S represent the solid-liquid interface and ablated surface positions, respectively; \bar{v}_l and \bar{v}_S are the velocities of these two interfaces, which are unknown for the ablation problem. The time when melting starts, t_0 , and the temperature distribution in the medium at the

onset of melting, $T_0(\bar{r}, t_0)$, are to be found from the solution of the pre-melt stage. On the other hand, t_v is the time when ablation begins; $T_L(\bar{r}, t_v)$ and $T_S(\bar{r}, t_v)$ are the temperature distributions in the liquid and solid regions, respectively, at the beginning of the ablation stage. Like before, t_v , $T_L(\bar{r}, t_v)$, and $T_S(\bar{r}, t_v)$ will be found from the melting stage solution.

The source-and-sink method is used to derive the temperature for the ablation problem. In this method, the solution will be sought in a fixed domain which is extended to cover AULUS; the ablation front is taken to be the second interface which separates the gas and liquid regions. The first interface appearing in the melting stage is the interface that separates the liquid and solid regions. As is previously the case, the ablated region is a fictitious region, while the solid and liquid regions are real. The interface conditions imposed at the melt front in the original ablation problem are still those in the new problem. However, the conditions imposed at the surface are now taken to be the conditions at the interface between the liquid and gas regions. The equivalent problem is thus worked in a fixed domain so that the surface of this domain is exposed to an imaginary heat flux whose magnitude is adjusted to meet the interface conditions specified. In this method, the interface locations and the fictitious heat flux on the fixed boundary are treated as unknowns. Once they are found, they can be used in the temperature equation to complete the solution. Following this approach, the ablation problem is recast into an equivalent problem as follows.

Equivalent problem

Governing equation--

$$\bar{\mathbf{r}} \in (A \cup L \cup S)$$

$$\nabla^2 T(\bar{\mathbf{r}}, t) - \frac{\rho L_f}{k} \bar{\mathbf{v}}_l \cdot \hat{\mathbf{n}}_l \delta_1(\bar{\mathbf{r}} - \bar{\mathbf{r}}_l) - \frac{\rho L_v}{k} \bar{\mathbf{v}}_S \cdot \hat{\mathbf{n}}_S \delta_2(\bar{\mathbf{r}} - \bar{\mathbf{r}}_S) = \frac{1}{\alpha} \frac{\partial T(\bar{\mathbf{r}}, t)}{\partial t}, \quad (5.18)$$

$$t > t_v$$

Initial condition--

$$T(\bar{\mathbf{r}}, t_v) = T_1(\bar{\mathbf{r}}, t_v) \quad (5.19)$$

Interface conditions--

$$G(t) = -k \frac{\partial T(\bar{\mathbf{r}}_S, t)}{\partial n_S} + \rho L_v \bar{\mathbf{v}}_S \cdot \hat{\mathbf{n}}_S \quad (5.20)$$

$$T(\bar{\mathbf{r}}_l, t) = T_m \quad (5.21)$$

$$T(\bar{\mathbf{r}}_S, t) = T_v \quad (5.22)$$

Here equation (5.18) can be reduced to equations (5.8) and (5.13) by using the definition of the Dirac delta function. Furthermore, by integrating (5.18) across $\bar{\mathbf{r}}_l$ and $\bar{\mathbf{r}}_S$, one obtains (5.9) and (5.16). Other equivalences for this stage are apparent. In (5.19), $T_1(\bar{\mathbf{r}}, t_v)$ is the temperature distribution in the medium at the onset of

ablation, which is found by solving the melting stage. Using equation (3.18), the melting stage solution can be expressed as

$$T_1(\bar{r}, t) = \int_{L \cup S} G(\bar{r}, t | \bar{r}', t_0) T_i(\bar{r}') dV' - \frac{L_f}{c} \int_{t_0}^t \int_{L \cup S} G(\bar{r}, t | \bar{r}', \tau) \bar{v}_l \cdot \hat{n}_l \delta(\bar{r}' - \bar{r}_l) dV' d\tau + \sum_i \left\{ \right\} \quad (5.23)$$

Note that the minus sign has been taken for the Dirac-delta-function term; \bar{r}_f and L in that equation have been changed to \bar{r}_l and L_f , respectively. This equation is used to derive the solid-liquid interface position by setting \bar{r} to \bar{r}_l and $T_1(\bar{r}_l, t)$ to T_m as

$$T_m = \int_{L \cup S} G(\bar{r}_l, t | \bar{r}', t_0) T_i(\bar{r}') dV' - \frac{L_f}{c} \int_{t_0}^t \int_{L \cup S} G(\bar{r}_l, t | \bar{r}', \tau) \bar{v}_l \cdot \hat{n}_l \delta(\bar{r}' - \bar{r}_l) dV' d\tau + \sum_i \left\{ \right\} \quad (5.24)$$

Also substituting t_v for t in (5.23) gives the temperature at the onset of ablation as

$$T_1(\bar{r}, t_v) = \int_{L \cup S} G(\bar{r}, t_v | \bar{r}', t_0) T_i(\bar{r}') dV' - \frac{L_f}{c} \int_{t_0}^{t_v} \int_{L \cup S} G(\bar{r}, t_v | \bar{r}', \tau) \bar{v}_l \cdot \hat{n}_l \delta(\bar{r}' - \bar{r}_l) dV' d\tau + \sum_i \left\{ \right\} \quad (5.25)$$

Changing \bar{r} to \bar{r}_0 and $T(\bar{r}_0, t_v)$ to T_v gives

$$T_v = \int_{L \cup S} G(\bar{r}_0, t_v | \bar{r}', t_0) T_i(\bar{r}') dV' - \frac{L_f}{c} \int_{t_0}^{t_v} \int_{L \cup S} G(\bar{r}_0, t_v | \bar{r}', \tau) \bar{v}_l \cdot \hat{n}_l \delta(\bar{r}' - \bar{r}_l) dV' d\tau + \sum_i \left\{ \right\} \quad (5.26)$$

which will be solved implicitly for the time when ablation starts (t_v).

In the ablation stage, the equivalent problem can be solved by again using equation (3.18) in which the domain of integration in the first and second terms on the right are changed to $A \cup L \cup S$. Also the Dirac delta function term is modified to include both solid-liquid and ablated interfaces; again minus signs are taken for the modified term, giving the result

$$\begin{aligned}
 T(\bar{r}, t) = & \int_{A \cup L \cup S} G(\bar{r}, t | \bar{r}', t_v) T_i(\bar{r}') dV' \\
 & - \frac{L}{c} \int_{t_v}^t \int_{A \cup L \cup S} G(\bar{r}, t | \bar{r}', \tau) \left\{ \bar{v}_l \cdot \hat{n}_l \delta(\bar{r}' - \bar{r}_l) + \bar{v}_s \cdot \hat{n}_s \delta(\bar{r}' - \bar{r}_s) \right\} dV' d\tau + \sum_i \left\{ \right\}
 \end{aligned} \tag{5.27}$$

Then by setting \bar{r} to \bar{r}_l and $T(\bar{r}_l, t)$ to T_m in this equation, there is derived

$$\begin{aligned}
 T_m = & \int_{A \cup L \cup S} G(\bar{r}_l, t | \bar{r}', t_0) T_i(\bar{r}') dV' \\
 & - \frac{L}{c} \int_{t_v}^t \int_{A \cup L \cup S} G(\bar{r}_l, t | \bar{r}', \tau) \left\{ \bar{v}_l \cdot \hat{n}_l \delta(\bar{r}' - \bar{r}_l) + \bar{v}_s \cdot \hat{n}_s \delta(\bar{r}' - \bar{r}_s) \right\} dV' d\tau + \sum_i \left\{ \right\}
 \end{aligned} \tag{5.28}$$

In a similar fashion, setting \bar{r} to \bar{r}_s and $T(\bar{r}_s, t)$ to T_v in (5.27) gives the following equation for T_v as

$$\begin{aligned}
T_v = & \int_{A \cup L \cup S} G(\bar{r}_S, t | \bar{r}', t_v) T_i(\bar{r}') dV' \\
& - \frac{L}{c} \int_{t_v}^t \int_{A \cup L \cup S} G(\bar{r}_S, t | \bar{r}', \tau) \left\{ \bar{v}_l \cdot \hat{n}_l \delta(\bar{r}' - \bar{r}_l) + \bar{v}_S \cdot \hat{n}_S \delta(\bar{r}' - \bar{r}_S) \right\} dV' d\tau + \sum_i \left\{ \right\}
\end{aligned} \tag{5.29}$$

Furthermore, equation (5.27) is differentiated and forced to satisfy (5.20) as

$$\begin{aligned}
G(t) = & -k \left\{ \int_{A \cup L \cup S} \frac{\partial G(\bar{r}_S, t | \bar{r}', t_v)}{\partial n_S} T_i(\bar{r}') dV' - \frac{L}{c} \int_{t_v}^t \int_{A \cup L \cup S} \frac{\partial G(\bar{r}_S, t | \bar{r}', \tau)}{\partial n_S} \left\{ \bar{v}_l \cdot \hat{n}_l \delta(\bar{r}' - \bar{r}_l) \right. \right. \\
& \left. \left. + \bar{v}_S \cdot \hat{n}_S \delta(\bar{r}' - \bar{r}_S) \right\} dV' d\tau + \sum_i \frac{\partial}{\partial n_S} \left\{ \right\}_{\bar{r}=\bar{r}_S} \right\} + \rho L_v \bar{v}_S \cdot \hat{n}_S
\end{aligned} \tag{5.30}$$

In the equations given above, the boundary condition is embodied in the summation term, which is to be taken from equations (4.15) and (4.16) in which t_0 and n_i are changed to t_v and n_S , respectively. Clearly, equations (5.28) through (5.30) contain three unknowns (\bar{v}_l , \bar{v}_S , and $g(\bar{r}'_0, t)$); there are three equations to solve them.

5.2 Examples

The general analysis given in the previous section is now used to solve example problems in a semi-infinite domain. Here three examples are given, and they are for a subcooled medium imposed with constant and time-variant heat-flux conditions.

The pre-melt stage can be solved by using (4.18) and (4.19). The melting stage can then be solved by using (5.23) as

$$\begin{aligned}
T_1(x,t) = T_i + \frac{1}{k\sqrt{\pi}} \int_{\tau=0}^t \frac{G(\tau)}{(t-\tau)^{1/2}} \exp\left[-\frac{x^2}{4\alpha(t-\tau)}\right] d\tau \\
- \frac{L_f}{c} \int_{\tau=t_0}^t \frac{dR_1(\tau)}{d\tau} \frac{1}{\sqrt{4\pi\alpha(t-\tau)}} \left\{ \exp\left(-\frac{(x-R_1(\tau))^2}{4\alpha(t-\tau)}\right) + \exp\left(-\frac{(x+R_1(\tau))^2}{4\alpha(t-\tau)}\right) \right\} d\tau
\end{aligned} \tag{5.31}$$

Correspondingly, the temperature distribution at t_v can be derived by using (5.25) as

$$\begin{aligned}
T_1(x,t_v) = T_i + \frac{1}{k\sqrt{\pi}} \int_{\tau=0}^{t_v} \frac{G(\tau)}{(t_v-\tau)^{1/2}} \exp\left[-\frac{x^2}{4\alpha(t_v-\tau)}\right] d\tau \\
- \frac{L_f}{c} \int_{\tau=t_0}^{t_v} \frac{dR_1(\tau)}{d\tau} \frac{1}{\sqrt{4\pi\alpha(t_v-\tau)}} \left\{ \exp\left(-\frac{(x-R_1(\tau))^2}{4\alpha(t_v-\tau)}\right) + \exp\left(-\frac{(x+R_1(\tau))^2}{4\alpha(t_v-\tau)}\right) \right\} d\tau
\end{aligned} \tag{5.32}$$

which is used as the initial condition for the ablation stage. Also using equation (5.26) gives

$$T_v = T_i + \frac{1}{k\sqrt{\pi}} \int_{\tau=0}^{t_v} \frac{G(\tau)}{(t_v-\tau)^{1/2}} d\tau - \frac{L_f}{c} \int_{\tau=t_0}^{t_v} \frac{dR_1(\tau)}{d\tau} \frac{1}{\sqrt{\pi\alpha(t_v-\tau)}} \exp\left(-\frac{R_1^2(\tau)}{4\alpha(t_v-\tau)}\right) d\tau \tag{5.33}$$

which is solved implicitly for t_v , the time when ablation starts.

The ablation-stage solution is derived by using (5.27), (5.28), (5.29) as

$$\begin{aligned}
T(x,t) = T_i + \frac{1}{k} \sqrt{\frac{\alpha}{\pi}} \int_{\tau=0}^{t_v} \frac{G(\tau)}{(t-\tau)^{1/2}} \exp\left[-\frac{x^2}{4\alpha(t-\tau)}\right] d\tau + \frac{1}{k} \sqrt{\frac{\alpha}{\pi}} \int_{\tau=t_v}^t \frac{g(\tau)}{(t-\tau)^{1/2}} \exp\left[-\frac{x^2}{4\alpha(t-\tau)}\right] d\tau \\
- \frac{L_f}{c} \int_{\tau=t_0}^t \frac{dR_1(\tau)}{d\tau} \frac{1}{\sqrt{4\pi\alpha(t-\tau)}} \left\{ \exp\left(-\frac{(x-R_1(\tau))^2}{4\alpha(t-\tau)}\right) + \exp\left(-\frac{(x+R_1(\tau))^2}{4\alpha(t-\tau)}\right) \right\} d\tau \\
- \frac{L_v}{c} \int_{\tau=t_v}^t \frac{dR_2(\tau)}{d\tau} \frac{1}{\sqrt{4\pi\alpha(t-\tau)}} \left\{ \exp\left(-\frac{(x-R_2(\tau))^2}{4\alpha(t-\tau)}\right) + \exp\left(-\frac{(x+R_2(\tau))^2}{4\alpha(t-\tau)}\right) \right\} d\tau
\end{aligned} \tag{5.34}$$

$$\begin{aligned}
T_m = T_i + \frac{1}{k} \sqrt{\frac{\alpha}{\pi}} \int_{\tau=0}^{t_v} \frac{G(\tau)}{(t-\tau)^{1/2}} \exp\left[-\frac{R_1^2(t)}{4\alpha(t-\tau)}\right] d\tau + \frac{1}{k} \sqrt{\frac{\alpha}{\pi}} \int_{\tau=t_v}^t \frac{g(\tau)}{(t-\tau)^{1/2}} \exp\left[-\frac{R_1^2(t)}{4\alpha(t-\tau)}\right] d\tau \\
- \frac{L_f}{c} \int_{\tau=t_0}^t \frac{dR_1(\tau)}{d\tau} \frac{1}{\sqrt{4\pi\alpha(t-\tau)}} \left\{ \exp\left(-\frac{(R_1(t)-R_1(\tau))^2}{4\alpha(t-\tau)}\right) + \exp\left(-\frac{(R_1(t)+R_1(\tau))^2}{4\alpha(t-\tau)}\right) \right\} d\tau \\
- \frac{L_v}{c} \int_{\tau=t_v}^t \frac{dR_2(\tau)}{d\tau} \frac{1}{\sqrt{4\pi\alpha(t-\tau)}} \left\{ \exp\left(-\frac{(R_1(t)-R_2(\tau))^2}{4\alpha(t-\tau)}\right) + \exp\left(-\frac{(R_1(t)+R_2(\tau))^2}{4\alpha(t-\tau)}\right) \right\} d\tau
\end{aligned} \tag{5.35}$$

$$\begin{aligned}
T_v = T_i + \frac{1}{k} \sqrt{\frac{\alpha}{\pi}} \int_{\tau=0}^{t_v} \frac{G(\tau)}{(t-\tau)^{1/2}} \exp\left[-\frac{R_2^2(t)}{4\alpha(t-\tau)}\right] d\tau + \frac{1}{k} \sqrt{\frac{\alpha}{\pi}} \int_{\tau=t_v}^t \frac{g(\tau)}{(t-\tau)^{1/2}} \exp\left[-\frac{R_2^2(t)}{4\alpha(t-\tau)}\right] d\tau \\
- \frac{L_f}{c} \int_{\tau=t_0}^t \frac{dR_1(\tau)}{d\tau} \frac{1}{\sqrt{4\pi\alpha(t-\tau)}} \left\{ \exp\left(-\frac{(R_2(t)-R_1(\tau))^2}{4\alpha(t-\tau)}\right) + \exp\left(-\frac{(R_2(t)+R_1(\tau))^2}{4\alpha(t-\tau)}\right) \right\} d\tau \\
- \frac{L_v}{c} \int_{\tau=t_v}^t \frac{dR_2(\tau)}{d\tau} \frac{1}{\sqrt{4\pi\alpha(t-\tau)}} \left\{ \exp\left(-\frac{(R_2(t)-R_2(\tau))^2}{4\alpha(t-\tau)}\right) + \exp\left(-\frac{(R_2(t)+R_2(\tau))^2}{4\alpha(t-\tau)}\right) \right\} d\tau
\end{aligned} \tag{5.36}$$

Finally, using (5.30) gives the flux condition as

$$\begin{aligned}
G(t) = & \frac{R_2(t)}{2\sqrt{\pi\alpha}} \left\{ \int_{\tau=0}^{t_v} \frac{G(\tau)}{(t-\tau)^{3/2}} \exp\left[-\frac{R_2^2(t)}{4\alpha(t-\tau)}\right] d\tau + \int_{\tau=t_v}^t \frac{g(\tau)}{(t-\tau)^{3/2}} \exp\left[-\frac{R_2^2(t)}{4\alpha(t-\tau)}\right] d\tau \right\} \\
& - \frac{\rho L_f}{4\sqrt{\pi\alpha}} \int_{\tau=t_0}^t \frac{dR_1(\tau)}{d\tau} \frac{1}{(t-\tau)^{3/2}} \left\{ (R_2(t) - R_1(\tau)) \exp\left(-\frac{(R_2(t) - R_1(\tau))^2}{4\alpha(t-\tau)}\right) \right. \\
& \left. + (R_2(t) + R_1(\tau)) \exp\left(-\frac{(R_2(t) + R_1(\tau))^2}{4\alpha(t-\tau)}\right) \right\} d\tau \\
& - \frac{\rho L_v}{4\sqrt{\pi\alpha}} \int_{\tau=t_v}^t \frac{dR_2(\tau)}{d\tau} \frac{1}{(t-\tau)^{3/2}} \left\{ (R_2(t) - R_2(\tau)) \exp\left(-\frac{(R_2(t) - R_2(\tau))^2}{4\alpha(t-\tau)}\right) \right. \\
& \left. + (R_2(t) + R_2(\tau)) \exp\left(-\frac{(R_2(t) + R_2(\tau))^2}{4\alpha(t-\tau)}\right) \right\} d\tau + \rho L_v \frac{dR_2(t)}{dt}
\end{aligned} \tag{5.37}$$

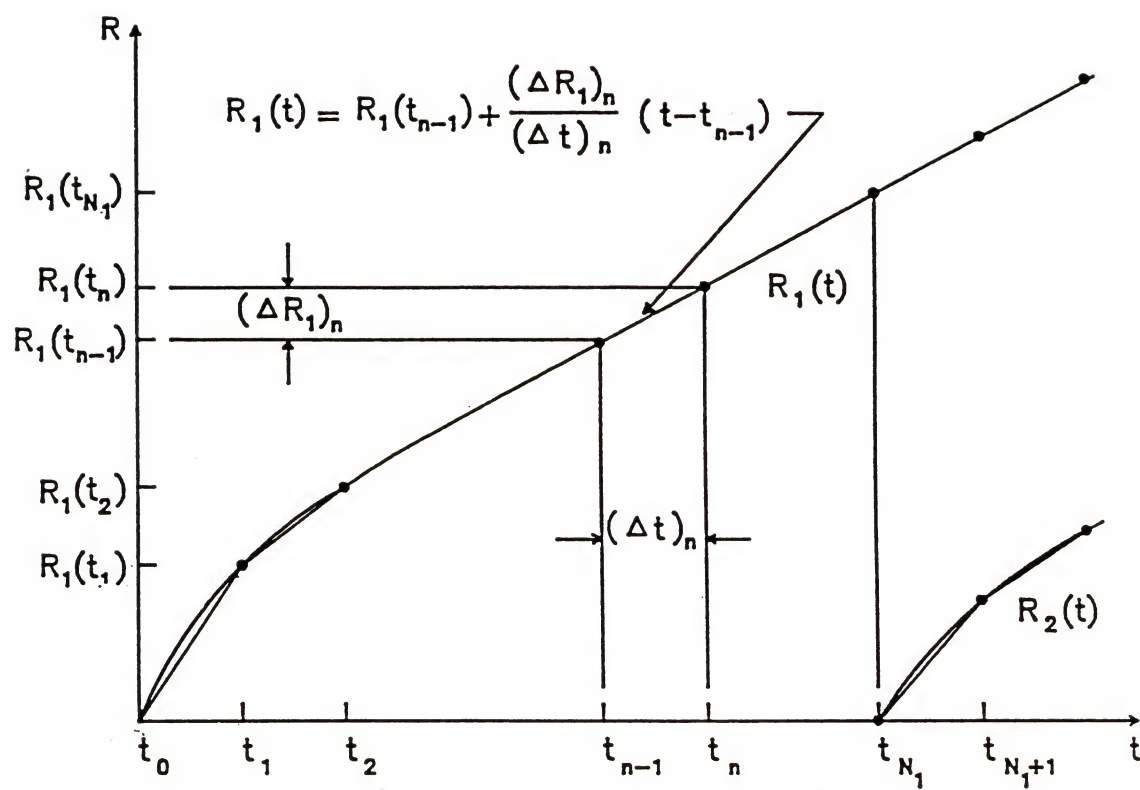
Equations (5.35), (5.36), and (5.37) contain three unknowns: R_1 , R_2 , and $g(t)$. They must be solved simultaneously by a numerical method as shown in the following section.

5.3 Numerical Solution of the Ablation Problem

A local linearization is again used for time (see Figure (5.2)), and the melting stage is solved first. In this effort, equation (5.24) is recast in a summation form as

$$\begin{aligned}
T_m = & T_i + \frac{1}{k\sqrt{\pi}} \int_{\tau=0}^t \frac{G(\tau)}{(t_{N_1} - \tau)^{1/2}} \exp\left[-\frac{R_1^2(t_{N_1})}{4\alpha(t_{N_1} - \tau)}\right] d\tau \\
& - \frac{L_f}{c} \sum_{n=1}^{N_1} \frac{dR_1(t_n)}{dt} \int_{\tau=t_{n-1}}^{t_n} \frac{1}{\sqrt{4\pi\alpha(t_{N_1} - \tau)}} \left\{ \exp\left(-\frac{(R_1(t_{N_1}) - R_1(\tau))^2}{4\alpha(t_{N_1} - \tau)}\right) \right. \\
& \left. + \exp\left(-\frac{(R_1(t_{N_1}) + R_1(\tau))^2}{4\alpha(t_{N_1} - \tau)}\right) \right\} d\tau
\end{aligned} \tag{5.38}$$

Figure 5.2 Linearization of solid-liquid interface and
 ablated surface position curves for the
 numerical solution



where $R_1(t_0)=0$; $N_1=1,2,\dots,N_1$; $t_{N_1} \leq t_v$. This equation is to be used to solve implicitly for t_v , the time when the ablation takes place.

Following the similar approach, (5.35), (5.36), (5.37) are linearized to be

$$\begin{aligned}
 T_m = & \frac{1}{k\sqrt{\pi}} \left\{ \int_{\tau=0}^{t_v} \frac{G(\tau)}{(t_{N_2} - \tau)^{1/2}} \exp\left[-\frac{R_1^2(t_{N_2})}{4\alpha(t_{N_2} - \tau)}\right] d\tau \right. \\
 & + \sum_{n=N_1+1}^{N_2} g(t_n) \int_{\tau=t_{n-1}}^{t_n} \frac{1}{(t_{N_2} - \tau)^{1/2}} \exp\left[-\frac{R_1^2(t_{N_2})}{4\alpha(t_{N_2} - \tau)}\right] d\tau \Big\} \\
 & - \frac{L_f}{c} \sum_{n=1}^{N_2} \frac{dR_1(t_n)}{dt} \int_{\tau=t_{n-1}}^{t_n} \frac{1}{\sqrt{4\pi\alpha(t_{N_2} - \tau)}} \cdot \\
 & \bullet \left\{ \exp\left(-\frac{(R_1(t_{N_2}) - R_1(\tau))^2}{4\alpha(t_{N_2} - \tau)}\right) + \exp\left(-\frac{(R_1(t_{N_2}) + R_1(\tau))^2}{4\alpha(t_{N_2} - \tau)}\right) \right\} d\tau \\
 & - \frac{L_v}{c} \sum_{n=N_1+1}^{N_2} \frac{dR_2(t_n)}{dt} \int_{\tau=t_{n-1}}^{t_n} \frac{1}{\sqrt{4\pi\alpha(t_{N_2} - \tau)}} \cdot \\
 & \bullet \left\{ \exp\left(-\frac{(R_1(t_{N_2}) - R_2(\tau))^2}{4\alpha(t_{N_2} - \tau)}\right) + \exp\left(-\frac{(R_1(t_{N_2}) + R_2(\tau))^2}{4\alpha(t_{N_2} - \tau)}\right) \right\} d\tau
 \end{aligned} \tag{5.39}$$

$$\begin{aligned}
 T_v = & \frac{1}{k\sqrt{\pi}} \left\{ \int_{\tau=0}^{t_v} \frac{G(\tau)}{(t_{N_2} - \tau)^{1/2}} \exp\left[-\frac{R_2^2(t_{N_2})}{4\alpha(t_{N_2} - \tau)}\right] d\tau \right. \\
 & + \sum_{n=N_1+1}^{N_2} g(t_n) \int_{\tau=t_{n-1}}^{t_n} \frac{1}{(t_{N_2} - \tau)^{1/2}} \exp\left[-\frac{R_2^2(t_{N_2})}{4\alpha(t_{N_2} - \tau)}\right] d\tau \Big\} \\
 & - \frac{L_f}{c} \sum_{n=1}^{N_2} \frac{dR_1(t_n)}{dt} \int_{\tau=t_{n-1}}^{t_n} \frac{1}{\sqrt{4\pi\alpha(t_{N_2} - \tau)}} \left\{ \exp\left(-\frac{(R_2(t_{N_2}) - R_1(\tau))^2}{4\alpha(t_{N_2} - \tau)}\right) \right. \\
 & + \exp\left(-\frac{(R_2(t_{N_2}) + R_1(\tau))^2}{4\alpha(t_{N_2} - \tau)}\right) \Big\} d\tau - \frac{L_v}{c} \sum_{n=N_1+1}^{N_2} \frac{dR_2(t_n)}{dt} \int_{\tau=t_{n-1}}^{t_n} \frac{1}{\sqrt{4\pi\alpha(t_{N_2} - \tau)}} \cdot \\
 & \bullet \left\{ \exp\left(-\frac{(R_2(t_{N_2}) - R_2(\tau))^2}{4\alpha(t_{N_2} - \tau)}\right) + \exp\left(-\frac{(R_2(t_{N_2}) + R_2(\tau))^2}{4\alpha(t_{N_2} - \tau)}\right) \right\} d\tau
 \end{aligned} \tag{5.40}$$

$$\begin{aligned}
G(t_{N_2}) = & \frac{R_2(t_{N_2})}{2\sqrt{\pi\alpha}} \left\{ \int_{\tau=0}^{t_v} \frac{G(\tau)}{(t_{N_2} - \tau)^{3/2}} \exp\left[-\frac{R_2^2(t_{N_2})}{4\alpha(t_{N_2} - \tau)}\right] d\tau \right. \\
& + \sum_{n=N_1+1}^{N_2} g(t_n) \int_{\tau=t_{n-1}}^{t_n} \frac{1}{(t_{N_2} - \tau)^{3/2}} \exp\left[-\frac{R_2^2(t_{N_2})}{4\alpha(t_{N_2} - \tau)}\right] d\tau \Big\} \\
& - \frac{\rho L_f}{4\sqrt{\pi\alpha}} \sum_{n=1}^{N_2} \frac{dR_1(t_n)}{d\tau} \int_{\tau=t_{n-1}}^{t_n} \frac{1}{(t_{N_2} - \tau)^{3/2}} \left\{ (R_2(t_{N_2}) - R_1(\tau)) \exp\left(-\frac{(R_2(t_{N_2}) - R_1(\tau))^2}{4\alpha(t_{N_2} - \tau)}\right) \right. \\
& + (R_2(t_{N_2}) + R_1(\tau)) \exp\left(-\frac{(R_2(t_{N_2}) + R_1(\tau))^2}{4\alpha(t_{N_2} - \tau)}\right) \Big\} d\tau \\
& - \frac{\rho L_v}{4\sqrt{\pi\alpha}} \sum_{n=N_1+1}^{N_2} \frac{dR_2(t_n)}{d\tau} \int_{\tau=t_{n-1}}^{t_n} \frac{1}{(t_{N_2} - \tau)^{3/2}} \left\{ (R_2(t_{N_2}) - R_2(\tau)) \cdot \right. \\
& \cdot \exp\left(-\frac{(R_2(t_{N_2}) - R_2(\tau))^2}{4\alpha(t_{N_2} - \tau)}\right) + (R_2(t_{N_2}) + R_2(\tau)) \exp\left(-\frac{(R_2(t_{N_2}) + R_2(\tau))^2}{4\alpha(t_{N_2} - \tau)}\right) \Big\} d\tau \\
& + \rho L_v \frac{dR_2(t_{N_2})}{dt}
\end{aligned} \tag{5.41}$$

where $R_2(t_{N_1})=0$; $N_2=N_1+1, N_1+2, \dots, N_2$; $t_v < t_{N_2}$.

A time marching scheme has been developed for the solution of these equations. In this effort, $R_1(t)$, $R_2(t)$, and $g(t)$ are solved simultaneously using (5.39), (5.40), and (5.41). Starting from t_{N_1+1} , they are solved incrementally step-by-step until the desired time is reached. A computer program (SSM) useful to solve the ablation problem with two moving boundaries is provided in Appendix G.

5.4 Numerical Solution of Temperature and Energy Storage

Once the interface positions and the condition ($R_1(t)$, $R_2(t)$, $g(t)$) are found numerically, they can be used in (5.34) to determine the temperature in the medium. In the absence of solutions for the combination problems in the literature, this temperature distribution is vital for checking the accuracy of the solution in the present investigation. Use is thus made of thermodynamic principles to derive the total heat consumed in heating, melting and vaporizing the medium as

$$Q_{act} = \rho \left\{ \int_0^{R_2(t)} [L_v + L_f + c(T_v - T_i)] dx + \int_{R_2(t)}^{R_1(t)} (L_f + c[T(x,t) - T_i]) dx + \int_{R_1(t)}^{\infty} c[T(x,t) - T_i] dx \right\} \quad (5.42)$$

The true heat input can be evaluated by using the imposed condition as

$$Q_{true} = \int_0^t G(t) dt \quad (5.43)$$

An overall error can then be defined as

$$E = 1 - \frac{Q_{act}}{Q_{true}} \quad (5.44)$$

Which embodies the errors not only in the temperature distribution but in the interface and boundary positions as well. This error can thus be safely taken to be the upper bound for the error in the solution.

5.5 Numerical Examples

The analysis devoted in the previous section was employed to obtain the numerical results for three examples imposed with constant, linear, and quadratic flux condition at the boundary of a subcooled medium; see Table 5.1. Again, aluminum has been used for tests; its properties are given in Table 3.2. The algorithm developed for solution of these examples. The pre-melt stage is solved first to yield t_0 , the time when melting starts, and $T_0(\bar{r}, t_0)$, which is the temperature in the medium at the onset of melting. This temperature is, in turn, used as the initial condition for the melting stage to solve for $R_1(t)$, the solid-liquid interface position, t_v , the time when ablation starts, and $T(\bar{r}, t_v)$, which is the temperature in the medium at the onset of the ablation. Finally, this last temperature is used as the initial condition for the ablation stage to solve for $R_1(t)$, $R_2(t)$. In this effort, the heat flux imposed at the boundary at x equal to zero is changed to $g(t)$ over the time interval (t_v, t) to account for the fact that, during that interval, the imposed heat flux at the fixed boundary is hypothetical in the sense that it is nonexistent physically. However, this flux is unknown; equations, (5.39), (5.40), and (5.41), must therefore be used to solve them ($R_1(t)$, $R_2(t)$, $g(t > t_v)$).

For a subcooled medium exposed to a large heat flux till vaporization, three phases appear in the medium. Examples 1, 2, and 3 address three situations, in which Example 1 is for a medium exposed to a constant heat-flux, Example 2 is for a linear heat-flux, and Example 3 is for a quadratic heat-flux (see Table 5.1).

Table 5.1 Conditions tested in three examples

Problem Description	Material	Heat Flux condition Imposed $G(t)$ (W/m^2); t (s)
1. Two-phase $T_i=300 \text{ K} < T_m$	Aluminum	$G(t)=5 \times 10^6$
2. Two-phase $T_i=300 \text{ K} < T_m$	Aluminum	$G(t)=3 \times 10^4 t$
3. Two-phase $T_i=300 \text{ K} < T_m$	Aluminum	$G(t)=2 \times 10^6 + 10^3 t^2$

Expectedly, in all these examples, the melt front appears first in the medium, and the ablation front follows sometime later. To show how these fronts move, Figures 5.3, 5.4, 5.5 are prepared for Examples 1, 2, and 3, respectively. It is interesting to note that, in all these figures, the melting front positions have very large curvature bended downward (see top curves in these figures), whereas for the ablation front positions all curves upward (see bottom curves) in duplication of what was seen earlier in Examples 3 and 4 in Chapter IV.

A temperature plot is also provided for the constant heat flux case as shown in Figure 5.6. Here three temperature curves are drawn for three different times covering the moment when vaporization starts and two instants of time when vaporization is in progress. To facilitate viewing the positions of the moving boundary as well as the solid-liquid interface, horizontal lines are drawn at the melting and vaporization temperatures. Thus the x-position of the intersection of the curves with the vaporization line locates the positions of the moving boundary, while that of the curves with the melting line locates the positions of the solid-liquid interface. Not given in this figure are temperature curves in the hypothetical regions for obvious reasons.

The numerical method developed for the solution of the combination problems in this chapter has also been tested for convergence and stability as shown in Figures, 5.7, 5.8, 5.9. They are good as shown in the figures. It should be noted that, for the numerical results presented in Figures 5.3 through 5.6, the time increment chosen is chosen to be 0.5 sec.

Figure 5.3

Trends of solid-liquid interface and ablated surface positions for a combination problem of subcooled medium imposed with a constant heat-flux condition

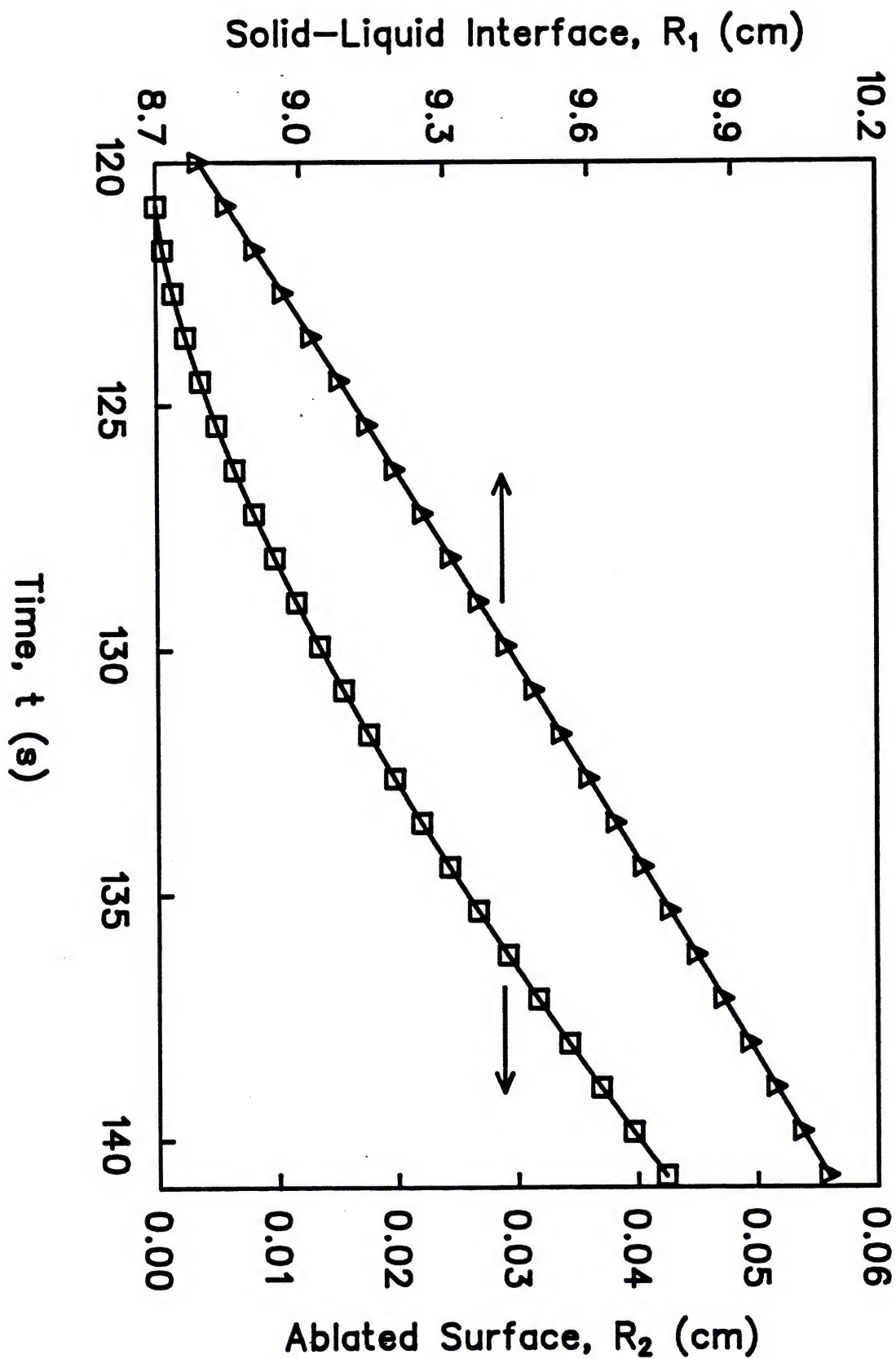


Figure 5.4

Trends of solid-liquid interface and ablated surface positions for a combination problem of subcooled medium imposed with a linear heat-flux condition

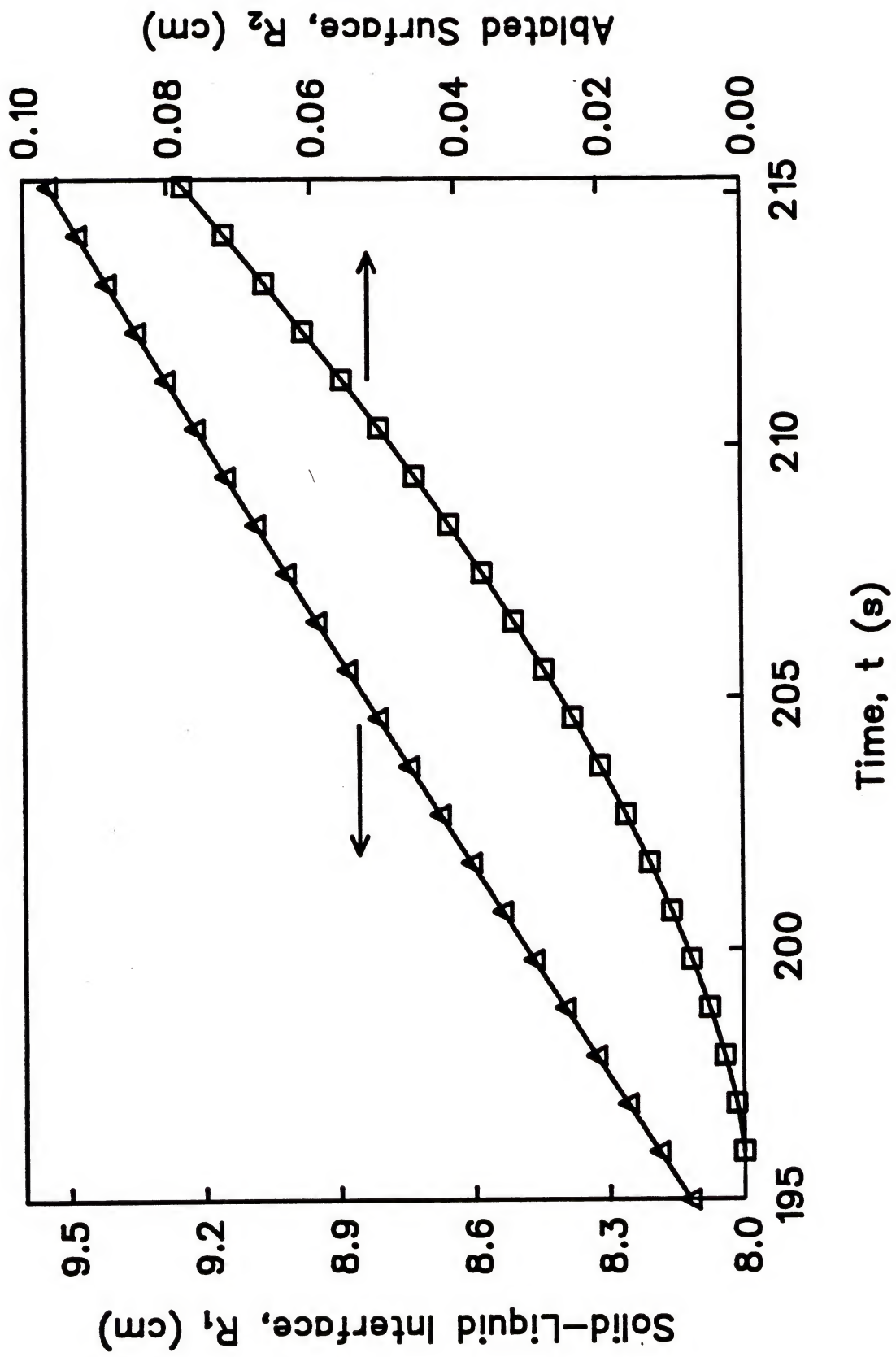


Figure 5.5

Trends of solid-liquid interface and ablated surface positions for a combination problem of subcooled medium imposed with a quadratic heat-flux condition

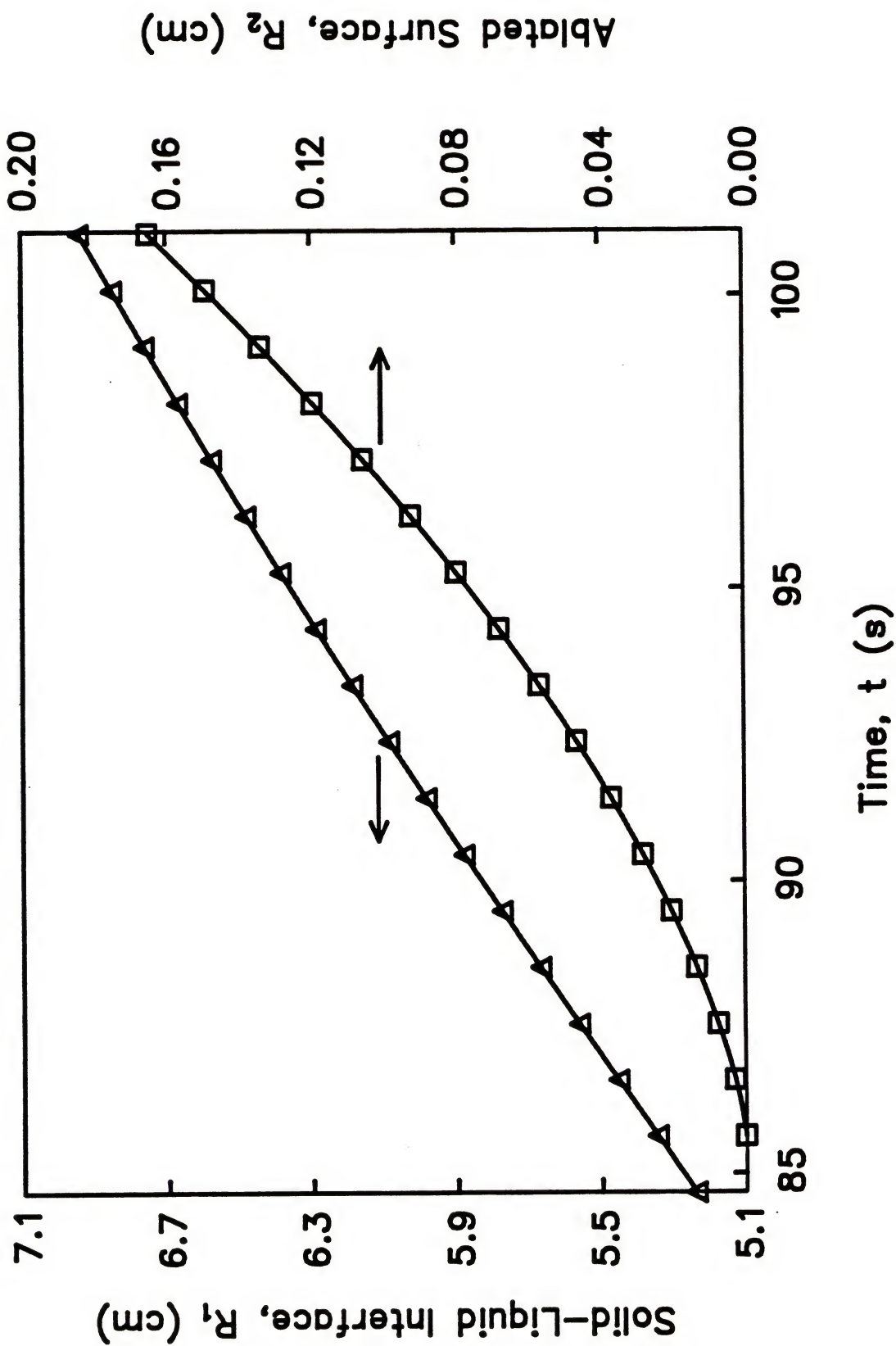


Figure 5.6 Temperature distributions in the medium at
different times during ablation for a combination
problem of subcooled medium imposed with a
constant heat-flux condition

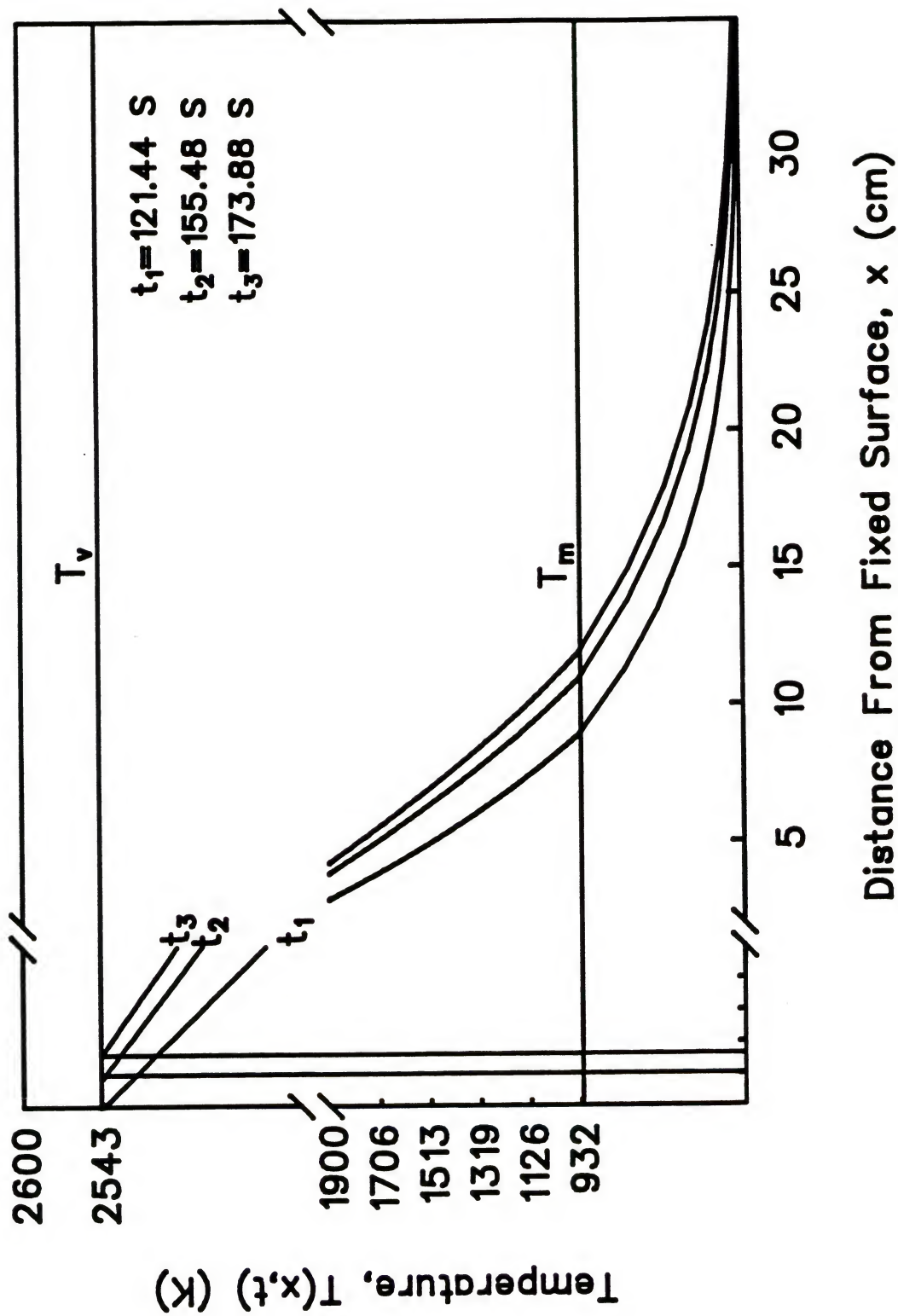


Figure 5.7 Stability and convergency test of the SSM in the solution of ablation for a combination problem of subcooled medium imposed with a constant heat-flux condition

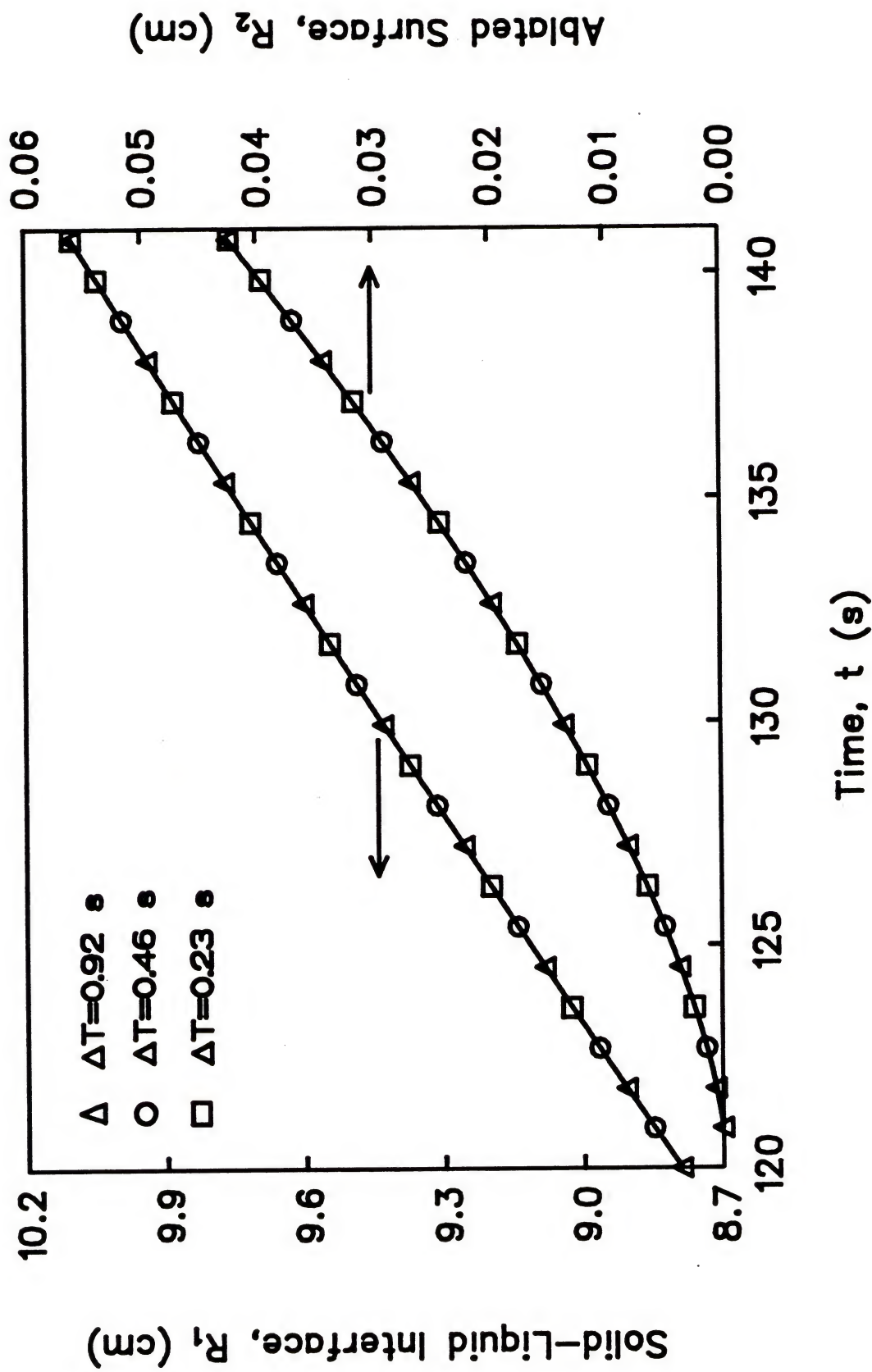


Figure 5.8 Stability and convergency test of the SSM in
the solution of ablation for a combination
problem of subcooled medium imposed with a
linear heat-flux condition

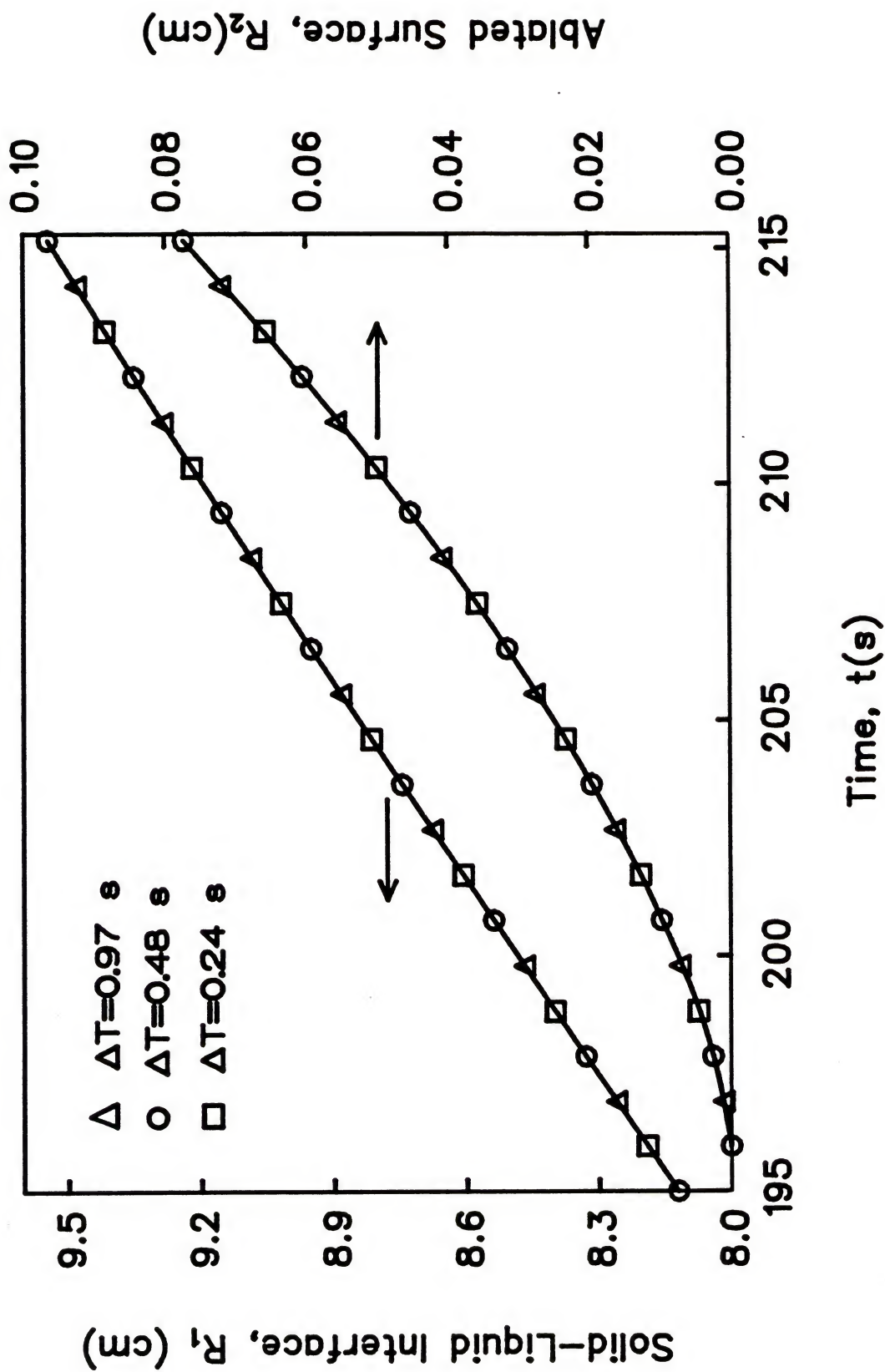
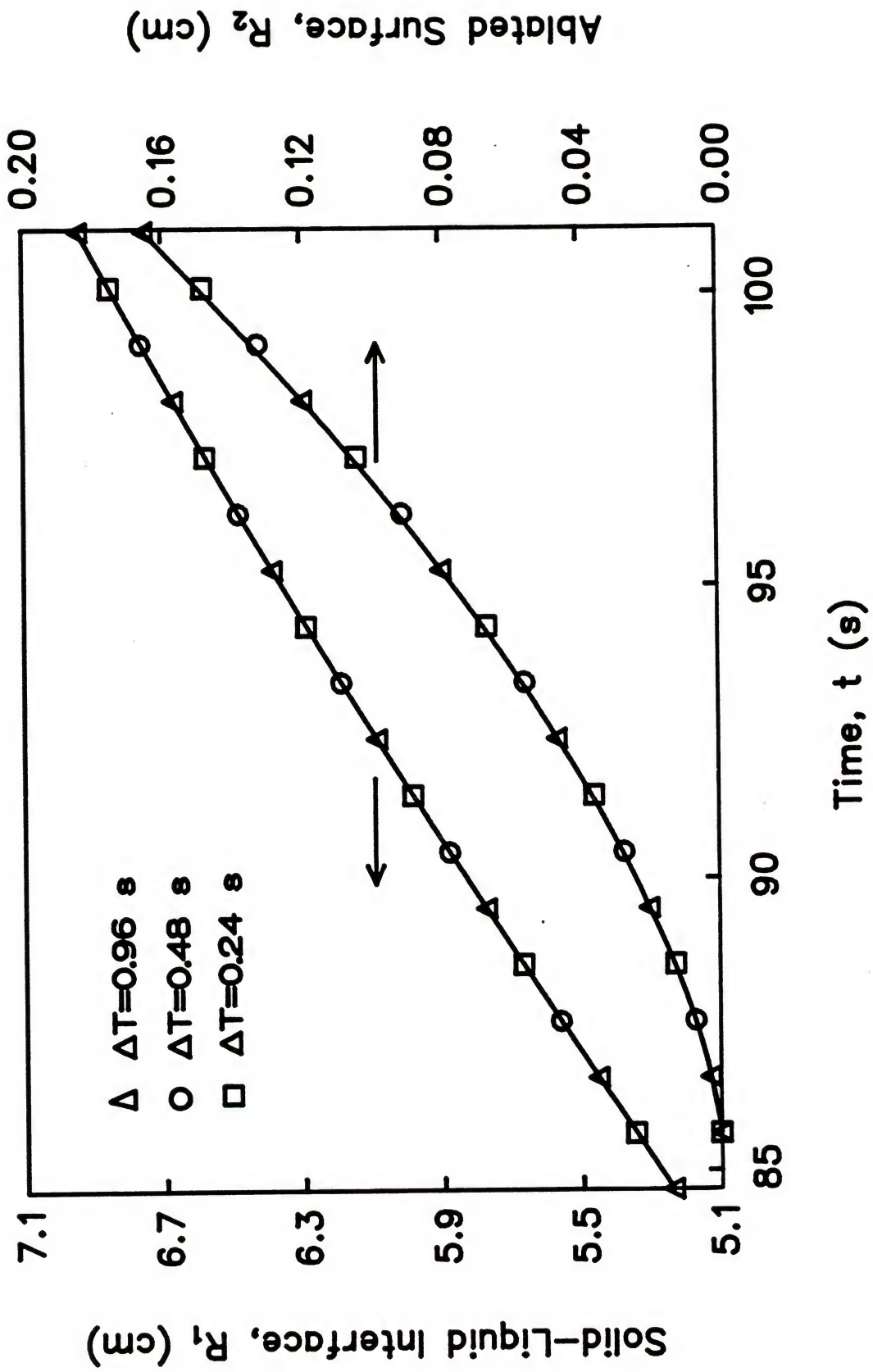


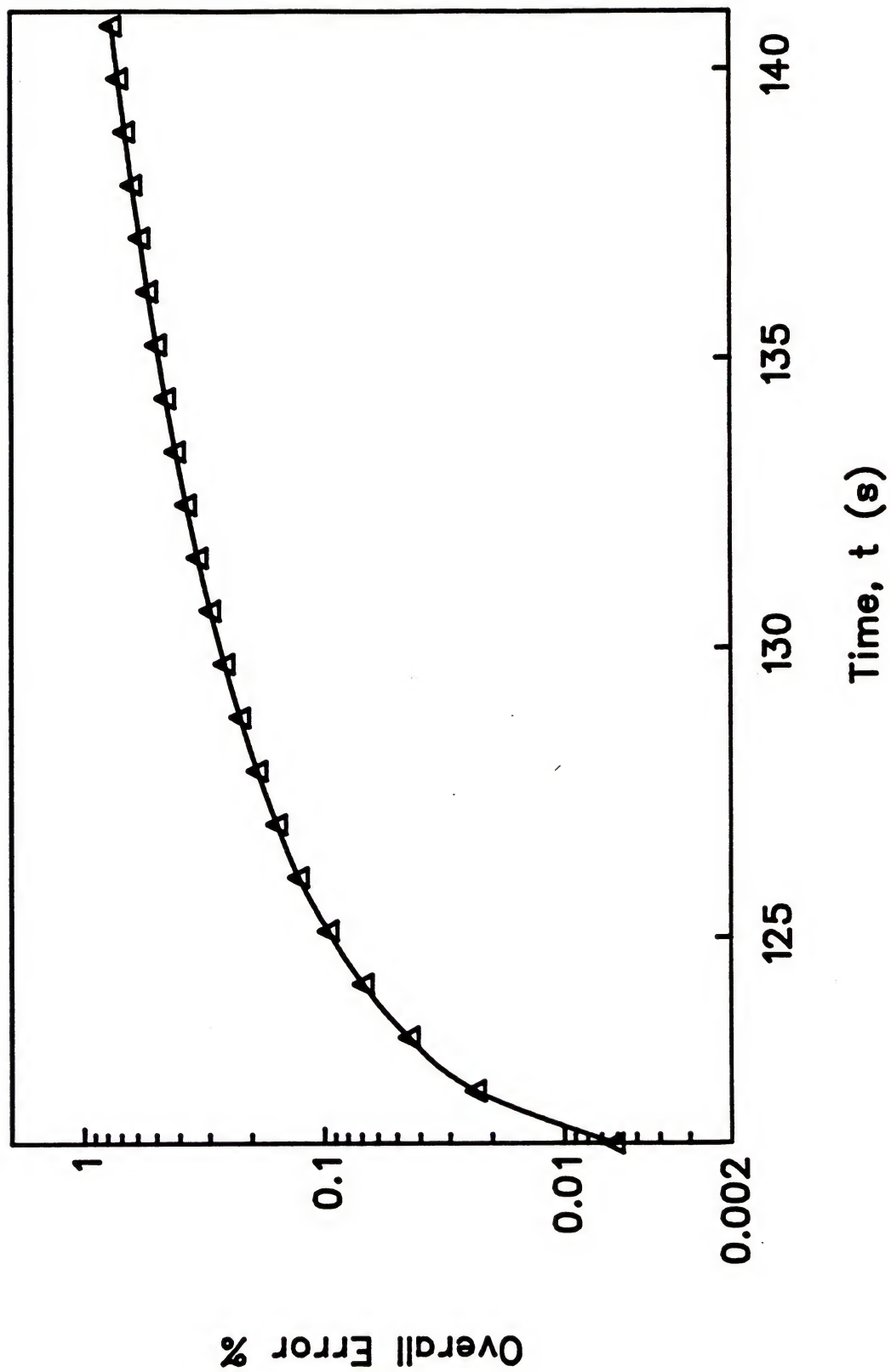
Figure 5.9 Stability and convergence test of the SSM in
the solution of ablation for a combination
problem of subcooled medium imposed with a
quadratic heat-flux condition



To the knowledge of the author, the combination problems solved in this chapter have rarely been attempted in the literature. Use is thus made of (5.44) to test the overall accuracy in the solution. In this regard, an error plot is prepared for example as shown in Figure 5.10. Here the overall error is shown to reach one percent asymptotically. However, it should be noted that (5.44) encompasses errors not only in the temperature distribution but in the boundary positions as well. Its being one percent in the figure provides renewed assurance of the overall accuracy in the work. It can also be taken as the upper bound for the error in the method.

Figure 5.10

Overall accuracy test of the SSM in the solution of the combination problem of subcooled medium imposed with a constant heat-flux condition



CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS

In the present study of the solution of phase change problems, it has been assumed that the thermophysical properties of the liquid and solid are constant and of equal value and convection and radiation effects are neglected. Based on the results obtained from the analysis, the following conclusions can be drawn:

1. The source and sink method has been developed for the solution of three phase-change problems, encompassing inverse Stefan problems, regular ablation problems, and the combination of ablation and Stefan problems. In each case, general solution methodologies are developed first, and they are applied to the solution of specific examples which are in semi-infinite space and imposed with either constant or time-variant temperature and flux conditions. The medium may be subcooled or superheated, but the properties are constant and of equal value in the liquid and solid phases.

2. Source-and-sink method has been used in the solution of all problems. In this method, a melting interface is taken to be a moving heat-sink front and a freezing interface is taken to be a moving heat-source front. Thus one set of equations is used for the solution of the temperature in all phase regions. In this effort, Green's functions are used and one temperature equation is derived.

Whether it is in the solid, liquid, or gas region depends on the position that is assigned in the equation.

3. For all the examples solved in this work, the interface positions are evaluated numerically. In this effort, a local linearization scheme is employed which divides the entire time range into small increments in which the interface velocities and the boundary condition (if unknown) are treated as constants. They can thus be taken out of their integrals, and the original convolution integrals are expressed in summations. This method is motivated by the fact that the interface position curves are usually a gradual function of time. They can thus be approximated as linear over small time increments. According to the present study, this local linearization only causes a slight error in the numerical solution of the interface position, which can still be reduced systematically by taking small time increments.

4. In the use of the source-and-sink method in the solution of the inverse Stefan problems, two approaches have been developed and they include the series expansion method and the time incremental method. In the series expansion method, the boundary condition is expanded in a power series, and the interface motion data at equal time intervals are used for input to solve for the coefficients in the series expansion. This power series is, in turn, used to generate the conditions at other times. On the other hand, in the time incremental approach, the interface motion data at consecutive times are used for input to determine the conditions imposed at the boundary incrementally. In either case, since the type of the condition imposed on the boundary is unknown a priori, a

method has been developed for determining the boundary temperature via the evaluated heat flux. Test results with eight examples indicate that the methods converge and are stable. Of the two methods developed, the time incremental approach is particularly attractive. Not only is the method more accurate, but also for the fact that the Stefan problems can be solved with this method by simple algorithms. Test results also suggest that the two-step solution of the flux then temperature may accumulate large error. In fact, the time saved in such a two-step operation is insignificant as compared with the separate, one-step evaluation of the temperature and heat flux. Solution of the inverse Stefan problems is successful.

5. The source-and-sink method has also been used in the solution of ablation problems and the combination of ablation and Stefan problems. In this application, the problems are solved in a fixed domain, and the condition that is originally imposed on the moving boundary is taken to be the condition imposed on the interior moving interface. Then by solving the motion of this interface together with the flux condition that is imposed on the fixed boundary, the temperature in the medium can be determined numerically. Seven examples have been provided that include one-, two-, and three-phase ablation for medium imposed with constant, linear, and quadratic flux conditions. The moving boundary positions have been compared with those evaluated with the methods documented in the literature. In all cases, the results are good. Errors are less than one percent, which include those in the

boundary position as well as in the integrated temperature in the medium. Solution of the ablation problems are also successful.

Following recommendation are made on the basis of the work done:

1. The analysis developed for the solution of inverse Stefan problems can be extended to the solution of inverse problems with multiple phases. For such problems, times for re-melt and re-freeze of the medium must be considered, and the analysis described in references 4 and 65 can be easily adapted for the solution of such problems. Also the problems solved in this work are in one-dimensional semi-infinite domain. Problems in finite domains can also be solved with the present method. For these problems, there will be two boundaries where two boundary conditions are imposed. Two unknowns must therefore be found simultaneously, and this requires the input of one additional condition at any interior point that is close to the boundary where no phase change takes place.

2. The thermal properties of the liquid and solid regions have been treated as constants and of equal value in this study. In practice, these properties may differ and may be a function of temperature. In these instances, double source and sink may be employed as suggested by Kolodner [17], and Kirchhoff transformation may be used for cases when the conductivity is a function of temperature. The property variations will be accounted for in future studies.

3. It will be of practical interest to solve the inverse Stefan problems and Stefan problems in two or three dimensions. It is expected that, because of the presence of the second (or third)

dimensions in these problems, the solution will be difficult. A hybrid analytic-finite difference scheme may be needed to solve such a problem.

APPENDIX A
EXTENSION OF THE SSM

For a Stefan problem in two-dimensional Cartesian system, the interface position can be represented as:

$$y_f = R(x_f, t)$$

Then, according to References 69 and 72:

$$v_n(t)\delta(\bar{r} - \bar{r}_f) = -\frac{\partial R}{\partial t} \delta(y - y_f)$$

Three-dimensional cases and problems in other coordinate systems can be formulated accordingly.

APPENDIX B
SSM FORTRAN PROGRAM FOR INVERSE STEFAN PROBLEM

```

C*****C
C      SOLUTION OF 1D INVERSE STEFAN PROBLEMS IN A SEMI-INFINITE      C
C      MEDIUM WITH OR WITHOUT SUBCOOLING BY THE SOURCE-AND-SINK      C
C      METHOD USING SERIES APPROACH                                    C
C                                                                    C
C      INPUT FILE UNIT NUMBER IS SET AT 12                            C
C      OUTPUT FILE UNIT IS SET AT 8 AND 16                            C
C                                                                    C
C              NOTATIONS                                             C
C                                                                    C
C      TO          =TIME WHEN PHASE CHANGE STARTS                    C
C      DT          =TIME-STEP SIZE                                   C
C      N           =NUMBER OF ENTRIES IN INPUT FILE                 C
C      ALPHA       =THERMAL DIFFUSIVITY                             C
C      CK          =THERMAL CONDUCTIVITY                             C
C      C           =SPECIFIC HEAT                                    C
C      CL          =LATENT HEAT OF FUSION                            C
C      RHO         =DENSITY                                           C
C      TI          =INITIAL TEMPERATURE                             C
C      TM          =MELTING TEMPERATURE                             C
C*****C

```

```

C
C-----MAIN PROGRAM
C

```

```

      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER (NP=250)
      DATA PI/3.14159265359D0/
      DATA RO/0.0D0/
      DIMENSION R(NP),UL(NP),CO(NP,NP),A(NP,NP),B1(NP)
      DIMENSION B3(NP),B23(NP),B2(NP,NP),X(NP),AA(NP,NP)
      DIMENSION GG(NP),GLL(NP),A23(NP),TEM(NP),B(NP)
      CHARACTER *40 FNAME
      CHARACTER*12 MQ
      COMMON/MINA/CC
      COMMON/MIN/FF
      COMMON/AL/ALPHA
      COMMON/ADT/DT
      COMMON/AAD/TO
      COMMON/MON/AAA
      COMMON/MONA/BB
      COMMON/EE/E
      COMMON/FF/F
      COMMON/ICC/IC
      COMMON/SAR/TT

```

```

COMMON/SARA/DD
COMMON/MON1/AA1
INTEGER RDSTAT,OPSTAT
EXTERNAL FUN,FUN1, DG, DG1
C
10 CONTINUE
PRINT*, 'ENTER FILE NAME OR MQ TO QUIT:'
READ '(A)', FNAME
OPEN(UNIT=1, FILE=FNAME, STATUS='OLD', IOSTAT=OPSTAT)
IF(.NOT.(OPSTAT.EQ. 0 .OR. FNAME.EQ. 'MQ')) GO TO 10
IF (FNAME.NE. 'MQ') THEN
C
PRINT*, 'DT, N, TO, TM, TI:'
READ*, DT, N, TO, TM, TI
READ(12,*) C, CL, CK, RHO
ALPHA=CK/RHO/C
COEF=C/CL/CK*DSQRT(ALPHA/PI)
C
C READ DATA FILE
C
DO I=1,N
    READ(UNIT=1, FMT=*, IOSTAT=RDSTAT) UL(I), R(I)
ENDDO
C
C FINDING VECTOR B1
C
DO I=1,N
    B1(I)=0.0D0
ENDDO
DO I=1,N
    B1(I)=C*TM/CL
ENDDO
PRINT*, 'B1', B1(1), B1(2)
C
C FINDING MATRIX B2
C
DO K=1,N
    DO I=1,N
        B2(I,K)=0.0D0
    ENDDO
ENDDO
C
CC=R0
TT=TO
CS=0.0D0
DO I=1,N
    AAA=UL(I)-TO
    FF=R(I)
    DO K=I,N
        BB=R(K)
        IF(K.EQ. 1) THEN
            CALL ROMBERG(DG, CS, AAA, ANTG)
        ELSE
            AA1=UL(K)-TO

```

```

        CALL ROMBERG(DG1,CS,AAA,ANTG)
    ENDIF
    B2(K,I)=ANTG
ENDDO
CS=UL(I)-T0
CC=R(I)
TT=UL(I)-T0
ENDDO
PRINT*, 'B2', ((B2(K,I), I=1,3), K=1,3)
C
C    FINDING VECTOR B3
C
    DO K=1,N
        B3(K)=0.0D0
    ENDDO
    DO K=1,N
        B3(K)=(R(K)-R(K-1))/DT
    ENDDO
    PRINT *, 'B3', B3(1), B3(2), B3(3)
C
C    FINDING MATRIX C0
C
    DO I=1,N
        DO J=1,N
            C0(I,J)=0.0D0
        ENDDO
    ENDDO
C
    T0=0.0D0
    DO J=1,N
        F=R(J)
        E=UL(J)
        DO I=1,J
            IC=I
            CALL ROMBERG(FUN,T0,E,COANT)
            C0(J,I)=COANT
        ENDDO
    ENDDO
    PRINT*, 'C0', ((A(I,J), J=1,3), I=1,3)
C
C    MULTIPLY MATRIX C0 BY THE COEFF.
C
    DO I=1,N
        DO J=1,N
            A(J,I)=0.0D0
        ENDDO
    ENDDO
    DO I=1,N
        DO J=1,N
            A(J,I)=C0(J,I)*COEF
        ENDDO
    ENDDO
    PRINT*, 'A', ((A(I,J), J=1,3), I=1,3)
C

```

```

C      MULTIPLY MATRIX B2 BY VECTOR B3
C
      DO I=1,N
        B23(I)=0.0D0
      ENDDO
      DO I=1,N
        DO J=1,N
          B23(I)=B23(I)+B2(I,J)*B3(J)
        ENDDO
      ENDDO
      PRINT*, 'B23', B23(1), B23(2)

C
C      FINDING VECTOR B
C
      DO I=1,N
        B(I)=0.0D0
      ENDDO
      DO I=1,N
        B(I)=B1(I)+B23(I)
      ENDDO
      DO I=1,N
        X(I)=B(I)
        DO J=1,N
          AA(I,J)=A(I,J)
        ENDDO
      ENDDO
      CALL LUDCMP(AA,N,NP,INDX,D)
      CALL LUBKSB(AA,N,NP,INDX,X)
      IDUM=-13
      DO I=1,N
        X(I)=X(I)*(1.0+0.2*RAN3(IDUM))
      ENDDO
      CALL MPROVE(A,AA,N,NP,INDX,B,X)
      ENDIF
      WRITE(16,*) ' AN'
      DO I=1,N
        WRITE(16,*) X(I)
      ENDDO
      DO I=1,20
        PP=X(N)
        DO J=N-1,1,-1
          PP=PP*I+X(J)
        ENDDO
        GG(I)=PP
        PRINT*,GG(I)
      ENDDO

C
C      FINDING TEMPERATURE
C
      DT=1
      DO I=1,NN
        READ(8,*)UL(I),R(I)
        PRINT*,R(I)
      ENDDO

```

```

DO I=1,NN
  DO J=1,NN
    A(I,J)=0.0D0
  ENDDO
ENDDO
IC=1.0D0
CSS=0.0D0
DO I=1,NN
  AAAA=UL(I)
  DO K=I,NN
    F=0.0D0
    E=UL(K)
    CALL ROMBERG(FUN,CSS,AAAA,COANT)
    A(K,I)=COANT
  ENDDO
  CSS=UL(I)
ENDDO

```

C

```

DO I=1,NN
  DO J=1,NN
    B2(I,J)=0.0D0
  ENDDO
ENDDO

```

C

```

CC=R0
TT=T0
CS=0.0D0
DO I=1,NN
  AAA=UL(I)-T0
  FF=R(I)
  DO K=I,NN
    BB=0.0D0
    IF(K.EQ.1)THEN
      CALL ROMBERG(DG,CS,AAA,ANTG)
    ELSE
      AA1=UL(K)-T0
      CALL ROMBERG(DG1,CS,AAA,ANTG)
    ENDIF
    B2(K,I)=ANTG
  ENDDO
  CS=UL(I)-T0
  CC=R(I)
  TT=UL(I)-T0
ENDDO

```

C

```

DO K=1,NN
  B3(K)=0.0D0
ENDDO
DO K=1,NN
  B3(K)=(R(K)-R(K-1))/DT
ENDDO

```

C

```

DO I=1,NN
  B23(I)=0.0D0

```

```

      ENDDO
      DO I=1,NN
        DO J=1,NN
          B23(I)=B23(I)+B2(I,J)*B3(J)
        ENDDO
      ENDDO
C
      DO I=1,NN
        A23(I)=0.0D0
      ENDDO
C
      DO I=1,NN
        DO J=1,NN
          A23(I)=A23(I)+A(I,J)*GG(J)
        ENDDO
      ENDDO
C
      DO I=1,NN
        TEM(I)=0.0D0
      ENDDO
      DO I=1,NN
        TEM(I)=1/CK*DSQRT(ALPHA/PI)*A23(I)-CL/C*B23(I)
        PRINT*,TEM(I)+TI
      ENDDO
C
      END
C*****C
C  SUBPROGRAM FUNCTION C
C*****C
      FUNCTION FUN(T)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON/AL/ALPHA
      COMMON/ICC/IC
      COMMON/EE/E
      COMMON/FF/F
      IF(E.EQ.T)THEN
        FUN=0.0D0
      ELSE
        FUN=T** (IC-1)/DSQRT(E-T)*DEXP(-(F**2)/4.D0/ALPHA/(E-T))
      ENDIF
      END
C
C*****C
C  SUBPROGRAM FUNCTION C
C*****C
      FUNCTION DG(T)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON/ADT/DT
      COMMON/AL/ALPHA
      COMMON/MON/AAA
      COMMON/MIN/FF
      COMMON/MONA/BB
      COMMON/AAD/TO
      DATA RO/0.D0/

```



```

DATA PI/3.14159265359D0/
IF(AAA.EQ.T) THEN
  DG=0.0D0
ELSE
  DG=1.D0/DSQRT(4.D0*PI*ALPHA*(AAA-T))*
& (DEXP(-(BB-FF/DT*T)**2/(4.D0*ALPHA*(AAA-T)))+
& DEXP(-(BB+FF/DT*T)**2/(4.D0*ALPHA*(AAA-T))))
ENDIF
END

C
C*****C
C      SUBPROGRAM FUNCTION                                C
C*****C
FUNCTION DG1(T)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
DATA PI/3.14159265359D0/
COMMON/ADT/DT
COMMON/AL/ALPHA
COMMON/MON/AAA
COMMON/MONA/BB
COMMON/AAD/TO
COMMON/SARA/DD
COMMON/MINA/CC
COMMON/MIN/FF
COMMON/SAR/TT
COMMON/MON1/AA1
IF(AA1.EQ.T) THEN
  DG1=0.0D0
ELSE
  DG1=1.D0/DSQRT(4*PI*ALPHA*(AA1-T))*
& (DEXP(-(BB-(CC+((FF-CC)/DT)*(T-TT)))*2)/(4.D0*ALPHA*(AA1-T)))+
& DEXP(-(BB+(CC+((FF-CC)/DT)*(T-TT)))*2)/(4.D0*ALPHA*(AA1-T)))
ENDIF
END

C
C*****C
C      SUBPROGRAM                                C
C      ROMBERG INTEGRATION                        C
C*****C
SUBROUTINE ROMBERG(FUNC,A,B,RESULT)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
EXTERNAL FUNC
PARAMETER(MAX=40,EPS=0.001D0)
DIMENSION T(MAX,MAX)

C
  T(1,1)=(B-A)*(FUNC(A)+FUNC(B))/2.0D0
  T(1,2)=T(1,1)/2.0D0+(B-A)*FUNC((A+B)/2.0D0)/2.0D0
  T(2,1)=(4.0D0*T(1,2)-T(1,1))/3.0D0
  J=3

C
C-----SUCCESSIVE APPLICATION OF TRAPEZOIDAL RULE
C
50  DELX=(B-A)/2.0D0**(J-1)
    X=A-DELX

```

```

      N=2**(J-2)
      SUM=0.0D0
      DO 100 I=1,N
          X=X+2.0D0*DELX
          SUM=SUM+FUNC(X)
100  CONTINUE
      T(1,J)=T(1,J-1)/2.0D0+DELX*SUM
C
C-----EXTRAPOLATION
C
      DO 200 L=2,J
          K=J+1-L
          T(L,K)=(4.0D0** (L-1)*T(L-1,K+1)-T(L-1,K))/
$          (4.0D0** (L-1)-1.0D0)
200  CONTINUE
C
C-----CHECK ACCURACY CRITERION
C
      IF(T(J,1) .EQ. 0.0D0) THEN
          RESULT=T(J,1)
          GO TO 111
      END IF
C
      IF(DABS((T(J,1)-T(J-1,1))/T(J,1)) .GE. EPS) THEN
          J=J+1
          IF(J.GT.MAX) THEN
              PAUSE 'TOO MANY STEPS'
              GO TO 111
          ELSE
              GO TO 50
          END IF
      ELSE
          RESULT=T(J,1)
      END IF
111  RETURN
      END
C
C*****C
C      SUBROUTINE MPROVE                                     C
C*****C
      SUBROUTINE MPROVE(A,ALUD,N,NP,INDX,B,X)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER (NMAX=100)
      DIMENSION A(NP,NP),ALUD(NP,NP),INDX(N),B(N),X(N),R(NMAX)
      REAL*8 SDP
      DO 12 I=1,N
          SDP=-B(I)
          DO 11 J=1,N
              SDP=SDP+DBLE(A(I,J))*DBLE(X(J))
11      CONTINUE
          R(I)=SDP
12  CONTINUE
      CALL LUBKSB(ALUD,N,NP,INDX,R)
      DO 13 I=1,N

```

```

      X(I)=X(I)-R(I)
13  CONTINUE
      RETURN
      END

```

```

C
C*****C
C  SUBROUTINE LU DECOMPOSITION  C
C*****C
      SUBROUTINE LUDCMP(A,N,NP,INDX,D)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER (NMAX=100,TINY=1.0E-20)
      DIMENSION A(NP,NP),INDX(N),VV(NMAX)
      D=1.0D0
      DO 12 I=1,N
          AAMAX=0.0D0
          DO 11 J=1,N
              IF (ABS(A(I,J)).GT.AAMAX) AAMAX=ABS(A(I,J))
11         CONTINUE
          IF (AAMAX.EQ.0.) PAUSE 'SINGULAR MATRIX.'
          VV(I)=1./AAMAX
12     CONTINUE
      DO 19 J=1,N
          IF (J.GT.1) THEN
              DO 14 I=1,J-1
                  SUM=A(I,J)
                  IF (I.GT.1) THEN
                      DO 13 K=1,I-1
                          SUM=SUM-A(I,K)*A(K,J)
13                     CONTINUE
                          A(I,J)=SUM
                      ENDIF
14                 CONTINUE
              ENDIF
              AAMAX=0.
              DO 16 I=J,N
                  SUM=A(I,J)
                  IF (J.GT.1) THEN
                      DO 15 K=1,J-1
                          SUM=SUM-A(I,K)*A(K,J)
15                     CONTINUE
                          A(I,J)=SUM
                      ENDIF
                      DUM=VV(I)*ABS(SUM)
                      IF (DUM.GE.AAMAX) THEN
                          IMAX=I
                          AAMAX=DUM
                      ENDIF
16                 CONTINUE
                  IF (J.NE.IMAX) THEN
                      DO 17 K=1,N
                          DUM=A(IMAX,K)
                          A(IMAX,K)=A(J,K)
                          A(J,K)=DUM
17                     CONTINUE

```

```

      D=-D
      VV(IMAX)=VV(J)
    ENDIF
    INDX(J)=IMAX
    IF(J.NE.N)THEN
      IF(A(J,J).EQ.0.)A(J,J)=TINY
      DUM=1./A(J,J)
      DO 18 I=J+1,N
        A(I,J)=A(I,J)*DUM
18      CONTINUE
    ENDIF
19  CONTINUE
    IF(A(N,N).EQ.0.)A(N,N)=TINY
    RETURN
  END

C
C*****C
C  SUBROUTINE BACKSUBSTITUTION                                C
C*****C
  SUBROUTINE LUBKSB(A,N,NP,INDX,B)
    IMPLICIT DOUBLE PRECISION(A-H,O-Z)
    DIMENSION A(NP,NP),INDX(N),B(N)
    II=0
    DO 12 I=1,N
      LL=INDX(I)
      SUM=B(LL)
      B(LL)=B(I)
      IF (II.NE.0)THEN
        DO 11 J=II,I-1
          SUM=SUM-A(I,J)*B(J)
11      CONTINUE
        ELSE IF (SUM.NE.0.) THEN
          II=I
        ENDIF
      B(I)=SUM
12  CONTINUE
      DO 14 I=N,1,-1
        SUM=B(I)
        IF(I.LT.N)THEN
          DO 13 J=I+1,N
            SUM=SUM-A(I,J)*B(J)
13      CONTINUE
        ENDIF
        B(I)=SUM/A(I,I)
14  CONTINUE
    RETURN
  END

C
C*****C
C  SUBROUTIN RANDOM                                          C
C*****C
  FUNCTION RAN3(IDUM)
    IMPLICIT DOUBLE PRECISION(A-H,O-Z)
    PARAMETER (MBIG=1000000000,MSEED=161803398,MZ=0,FAC=1.E-9)

```

```

DIMENSION MA(55)
DATA IFF /0/
IF(IDUM.LT.0.OR.IFF.EQ.0)THEN
  IFF=1
  MJ=MSEED-IABS(IDUM)
  MJ=MOD(MJ,MBIG)
  MA(55)=MJ
  MK=1
  DO 11 I=1,54
    II=MOD(21*I,55)
    MA(II)=MK
    MK=MJ-MK
    IF(MK.LT.MZ)MK=MK+MBIG
    MJ=MA(II)
11  CONTINUE
  DO 13 K=1,4
    DO 12 I=1,55
      MA(I)=MA(I)-MA(1+MOD(I+30,55))
      IF(MA(I).LT.MZ)MA(I)=MA(I)+MBIG
12  CONTINUE
13  CONTINUE
  INEXT=0
  INEXTP=31
  IDUM=1
ENDIF
INEXT=INEXT+1
IF(INEXT.EQ.56)INEXT=1
INEXTP=INEXTP+1
IF(INEXTP.EQ.56)INEXTP=1
MJ=MA(INEXT)-MA(INEXTP)
IF(MJ.LT.MZ)MJ=MJ+MBIG
MA(INEXT)=MJ
RAN3=MJ*FAC
RETURN
END

```

APPENDIX C
SSM FORTRAN PROGRAM FOR INVERSE STEFAN PROBLEM

```

C*****C
C      SOLUTION OF 1D INVERSE STEFAN PROBLEMS IN A SEMI-INFINITE      C
C      MEDIUM WITH OR WITHOUT SUBCOOLING BY THE SOURCE-AND-SINK      C
C      METHOD USING INCREMENTAL APPROACH                               C
C                                                                      C
C      INPUT FILE UNIT NUMBER IS SET AT 12                           C
C      OUTPUT FILE UNIT IS SET AT 8 AND 16                           C
C                                                                      C
C              NOTATIONS                                             C
C                                                                      C
C      TO          =TIME WHEN PHASE CHANGE STARTS                    C
C      DT          =TIME-STEP SIZE                                   C
C      N           =NUMBER OF ENTRIES IN INPUT FILE                  C
C      ALPHA       =THERMAL DIFFUSIVITY                             C
C      CK          =THERMAL CONDUCTIVITY                             C
C      C           =SPECIFIC HEAT                                    C
C      CL          =LATENT HEAT OF FUSION                             C
C      RHO         =DENSITY                                           C
C      TI          =INITIAL TEMPERATURE                             C
C      TM          =MELTING TEMPERATURE                             C
C*****C
C
C-----MAIN PROGRAM
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER (NP=250)
      DATA PI/3.14159265359D0/
      DATA RO/0.0D0/
      DIMENSION R(NP),UL(NP),CO(NP,NP),A(NP,NP)
      DIMENSION B3(NP),B23(NP),B2(NP,NP),X(NP)
      DIMENSION GG(NP),GLL(NP),A23(NP),TEM(NP)
      CHARACTER *40 FNAME
      CHARACTER*12 MQ
      COMMON/MINA/CC
      COMMON/MIN/FF
      COMMON/AL/ALPHA
      COMMON/ADT/DT
      COMMON/AAD/TO
      COMMON/MON/AAA
      COMMON/MONA/BB
      COMMON/EE/E
      COMMON/FF/F
      COMMON/ICC/IC
      COMMON/SAR/TT
      COMMON/SARA/DD

```



```

COMMON/MON1/AA1
INTEGER RDSTAT,OPSTAT
EXTERNAL FUN,FUN1, DG, DG1
C
10 CONTINUE
PRINT*, 'ENTER FILE NAME OR MQ TO QUIT:'
READ '(A)', FNAME
OPEN(UNIT=1, FILE=FNAME, STATUS='OLD', IOSTAT=OPSTAT)
IF(.NOT.(OPSTAT.EQ. 0 .OR. FNAME.EQ. 'MQ')) GO TO 10
IF (FNAME.NE. 'MQ') THEN
C
PRINT*, 'DT, N, TO, TM, TI:'
READ*, DT, N, TO, TM, TI
READ(12,*)C, CL, CK, RHO
ALPHA=CK/RHO/C
COEF=CL*CK/C*DSQRT(PI/ALPHA)
COEF1=CK*TM*DSQRT(PI/ALPHA)
C
C READ DATA FILE
C
DO I=1,N
  READ(UNIT=1, FMT=*, IOSTAT=RDSTAT) UL(I), R(I)
ENDDO
C
C FINDING MATRIX B2
C
DO K=1,N
  DO I=1,N
    B2(I,K)=0.0D0
  ENDDO
ENDDO
C
CC=R0
TT=T0
CS=0.0D0
DO I=1,N
  AAA=UL(I)-T0
  FF=R(I)
  DO K=I,N
    BB=R(K)
    IF(K.EQ. 1) THEN
      CALL ROMBERG(DG, CS, AAA, ANTG)
    ELSE
      AA1=UL(K)-T0
      CALL ROMBERG(DG1, CS, AAA, ANTG)
    ENDIF
    B2(K,I)=ANTG
  ENDDO
  CS=UL(I)-T0
  CC=R(I)
  TT=UL(I)-T0
ENDDO
PRINT*, 'B2', ((B2(K,I), I=1,3), K=1,3)
C

```

```

C   FINDING VECTOR B3
C
DO  K=1,N
    B3(K)=0.0D0
ENDDO
DO  K=1,N
    B3(K)=(R(K)-R(K-1))/DT
ENDDO
PRINT *, 'B3', B3(1), B3(2), B3(3)

C
C   FINDING MATRIX A
C
DO  I=1,N
    DO  J=1,N
        A(I,J)=0.0D0
    ENDDO
ENDDO

C
CSS=0.0D0
DO  I=1,N
    AAAA=UL(I)
    DO  K=I,N
        F=R(K)
        E=UL(K)
        CALL ROMBERG(FUN, CSS, AAAA, COANT)
        A(K,I)=COANT
    ENDDO
    CSS=UL(I)
ENDDO
PRINT*, 'A', ((A(I,J), J=1,3), I=1,3)

C
C   MULTIPLY MATRIX B2 BY VECTOR B3
C
DO  I=1,N
    B23(I)=0.0D0
ENDDO
DO  I=1,N
    DO  J=1,N
        B23(I)=B23(I)+B2(I,J)*B3(J)
    ENDDO
ENDDO
PRINT*, 'B23', B23(1), B23(2)

C
GG(1)=(COEF1+COEF*B23(1))/A(1,1)
XX=0.0D0
DO  I=2,N
    DO  K=1, I-1
        GLL(I)=GG(K)*A(I,K)+XX
        XX=GLL(I)
    ENDDO
    GG(I)=(COEF1+COEF*B23(I)-GLL(I))/A(I,I)
    XX=0.0D0
ENDDO
WRITE(10,*) ' G(I)'

```



```

DO I=1,N
  WRITE(10,*) GG(I)
ENDDO

```

C

```

DT=1
DO I=1,N
  DO J=1,N
    A(I,J)=0.0D0
  ENDDO
ENDDO
CSS=0.0D0
DO I=1,N
  AAAA=UL(I)
  DO K=I,N
    F=0.0D0
    E=UL(K)
    CALL ROMBERG(FUN,CSS,AAAA,COANT)
    A(K,I)=COANT
  ENDDO
  CSS=UL(I)
ENDDO

```

C

```

DO I=1,N
  DO J=1,N
    B2(I,J)=0.0D0
  ENDDO
ENDDO

```

C

```

CC=R0
TT=T0
CS=0.0D0
DO I=1,N
  AAA=UL(I)-T0
  FF=R(I)
  DO K=I,N
    BB=0.0D0
    IF(K.EQ.1)THEN
      CALL ROMBERG(DG,CS,AAA,ANTG)
    ELSE
      AA1=UL(K)-T0
      CALL ROMBERG(DG1,CS,AAA,ANTG)
    ENDIF
    B2(K,I)=ANTG
  ENDDO
  CS=UL(I)-T0
  CC=R(I)
  TT=UL(I)-T0
ENDDO

```

C

```

DO K=1,N
  B3(K)=0.0D0
ENDDO
DO K=1,N
  B3(K)=(R(K)-R(K-1))/DT

```

```

      ENDDO
C
      DO I=1,N
        B23(I)=0.0D0
      ENDDO
      DO I=1,N
        DO J=1,N
          B23(I)=B23(I)+B2(I,J)*B3(J)
        ENDDO
      ENDDO
C
      DO I=1,N
        A23(I)=0.0D0
      ENDDO
C
      DO I=1,N
        DO J=1,N
          A23(I)=A23(I)+A(I,J)*GG(J)
        ENDDO
      ENDDO
C
      DO I=1,N
        TEM(I)=0.0D0
      ENDDO
      DO I=1,N
        TEM(I)=1/CK*DSQRT(ALPHA/PI)*A23(I)-CL/C*B23(I)
        PRINT*,TEM(I)+TI
      ENDDO
    ENDIF
  END
C
C*****C
C  SUBPROGRAM FUNCTION C
C*****C
      FUNCTION FUN(T)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON/AL/ALPHA
      COMMON/EE/E
      COMMON/FF/F
      COMMON/ICC/IC
      IF(E.EQ.T)THEN
        FUN=0.0D0
      ELSE
        FUN=1.0D0/DSQRT(E-T)*DEXP(-(F**2)/4.D0/ALPHA/(E-T))
      ENDIF
    END
C
C
C*****C
C  SUBPROGRAM FUNCTION C
C*****C
      FUNCTION DG(T)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON/ADT/DT

```

```

COMMON/AL/ALPHA
COMMON/MON/AAA
COMMON/MIN/FF
COMMON/MONA/BB
COMMON/AAD/TO
DATA R0/0.D0/
DATA PI/3.14159265359D0/
IF(AAA.EQ.T) THEN
    DG=0.0D0
ELSE
    DG=1.D0/DSQRT(4.D0*PI*ALPHA*(AAA-T))*
&    (DEXP(-(BB-FF/DT*T)**2/(4.D0*ALPHA*(AAA-T)))+
&    DEXP(-(BB+FF/DT*T)**2/(4.D0*ALPHA*(AAA-T))))
ENDIF
END

C
C
C*****C
C    SUBPROGRAM FUNCTION                                C
C*****C
    FUNCTION DG1(T)
    IMPLICIT DOUBLE PRECISION(A-H,O-Z)
    DATA PI/3.14159265359D0/
    COMMON/ADT/DT
    COMMON/AL/ALPHA
    COMMON/MON/AAA
    COMMON/MONA/BB
    COMMON/AAD/TO
    COMMON/SARA/DD
    COMMON/MINA/CC
    COMMON/MIN/FF
    COMMON/SAR/TT
    COMMON/MON1/AA1
    IF(AA1.EQ.T)THEN
        DG1=0.0D0
    ELSE
        DG1=1.D0/DSQRT(4*PI*ALPHA*(AA1-T))*
&    (DEXP(-(BB-(CC+((FF-CC)/DT)*(T-TT)))**2)/(4.D0*ALPHA*(AA1-T)))+
&    DEXP(-(BB+(CC+((FF-CC)/DT)*(T-TT)))**2)/(4.D0*ALPHA*(AA1-T)))
    ENDIF
    END

C
C
C*****C
C    SUBROUTINE                                          C
C    ROMBERG INTEGRATION                                C
C*****C
    SUBROUTINE ROMBERG(FUNC,A,B,RESULT)
    IMPLICIT DOUBLE PRECISION(A-H,O-Z)
    EXTERNAL FUNC
    PARAMETER(MAX=50,EPS=0.001D0)
    DIMENSION T(MAX,MAX)

C
    T(1,1)=(B-A)*(FUNC(A)+FUNC(B))/2.0D0

```

```

T(1,2)=T(1,1)/2.0D0+(B-A)*FUNC((A+B)/2.0D0)/2.0D0
T(2,1)=(4.0D0*T(1,2)-T(1,1))/3.0D0
J=3

```

C

C-----SUCCESSIVE APPLICATION OF TRAPEZOIDAL RULE

C

```

50  DELX=(B-A)/2.0D0**(J-1)
    X=A-DELX
    N=2**(J-2)
    SUM=0.0D0
    DO 100 I=1,N
        X=X+2.0D0*DELX
        SUM=SUM+FUNC(X)
100  CONTINUE
    T(1,J)=T(1,J-1)/2.0D0+DELX*SUM

```

C

C-----EXTRAPOLATION

C

```

    DO 200 L=2,J
        K=J+1-L
        T(L,K)=(4.0D0**(L-1)*T(L-1,K+1)-T(L-1,K))/
$      (4.0D0**(L-1)-1.0D0)
200  CONTINUE

```

C

C-----CHECK ACCURACY CRITERION

C

```

    IF(T(J,1) .EQ. 0.0D0) THEN
        RESULT=T(J,1)
        GO TO 111
    END IF

```

C

```

    IF(DABS((T(J,1)-T(J-1,1))/T(J,1)) .GE. EPS) THEN
        J=J+1
        IF(J.GT.MAX) THEN
            PAUSE 'TOO MANY STEPS'
            GO TO 111
        ELSE
            GO TO 50
        END IF
    ELSE
        RESULT=T(J,1)
    END IF
111  RETURN
    END

```

APPENDIX D
SSM FORTRAN PROGRAM FOR INVERSE STEFAN PROBLEM

```

C*****C
C      SOLUTION OF 1D INVERSE STEFAN PROBLEMS IN A SEMI-INFINITE      C
C      MEDIUM WITH OR WITHOUT SUBCOOLING BY THE SOURCE-AND-SINK      C
C      METHOD USING SERIES APPROACH.  TEMPERATURE BOUNDARY IS        C
C      RETRIEVED VIA THE ASSUMED FLUX CONDITION.                      C
C                                                                      C
C      INPUT FILE UNIT NUMBER IS SET AT 12                          C
C      OUTPUT FILE UNIT IS SET AT 8 AND 16                          C
C                                                                      C
C              NOTATIONS                                           C
C                                                                      C
C      TO          =TIME WHEN PHASE CHANGE STARTS                  C
C      DT          =TIME-STEP SIZE                                C
C      N           =NUMBER OF ENTRIES IN INPUT FILE                C
C      ALPHA       =THERMAL DIFFUSIVITY                            C
C      CK          =THERMAL CONDUCTIVITY                           C
C      C           =SPECIFIC HEAT                                  C
C      CL          =LATENT HEAT OF FUSION                           C
C      RHO         =DENSITY                                         C
C      TI          =INITIAL TEMPERATURE                             C
C      TM          =MELTING TEMPERATURE                             C
C*****C
C
C-----MAIN PROGRAM
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER (NP=250)
      DATA PI/3.14159265359D0/
      DATA RO/0.0D0/
      DIMENSION  R(NP),UL(NP),CO(NP,NP),A(NP,NP),B1(NP)
      DIMENSION B3(NP),B23(NP),B2(NP,NP),X(NP),AA(NP,NP)
      DIMENSION GG(NP),GLL(NP),A23(NP),TEM(NP),B(NP)
      CHARACTER *40 FNAME
      CHARACTER*12 MQ
      COMMON/MINA/CC
      COMMON/MIN/FF
      COMMON/AL/ALPHA
      COMMON/ADT/DT
      COMMON/AAD/TO
      COMMON/MON/AAA
      COMMON/MONA/BB
      COMMON/EE/E
      COMMON/FF/F
      COMMON/ICC/IC
      COMMON/SAR/TT

```

```

COMMON/SARA/DD
COMMON/MON1/AA1
INTEGER RDSTAT,OPSTAT
EXTERNAL FUN,FUN1, DG, DG1
C
10 CONTINUE
PRINT*, 'ENTER FILE NAME OR MQ TO QUIT:'
READ '(A)', FNAME
OPEN(UNIT=1, FILE=FNAME, STATUS='OLD', IOSTAT=OPSTAT)
IF(.NOT.(OPSTAT .EQ. 0 .OR. FNAME .EQ. 'MQ')) GO TO 10
IF (FNAME .NE. 'MQ') THEN
C
PRINT*, 'DT, N, TO, TM, TI:'
READ*, DT, N, TO, TM, TI
READ(12,*)C, CL, CK, RH0
ALPHA=CK/RH0/C
COEF=C/CL/CK*DSQRT(ALPHA/PI)
C
C READ DATA FILE
C
DO I=1,N
    READ(UNIT=1, FMT=*, IOSTAT=RDSTAT) UL(I), R(I)
ENDDO
C
C FINDING VECTOR B1
C
DO I=1,N
    B1(I)=0.0D0
ENDDO
DO I=1,N
    B1(I)=C*TM/CL
ENDDO
PRINT*, 'B1', B1(1), B1(2)
C
C FINDING MATRIX B2
C
DO K=1,N
    DO I=1,N
        B2(I,K)=0.0D0
    ENDDO
ENDDO
C
CC=R0
TT=T0
CS=0.0D0
DO I=1,N
    AAA=UL(I)-T0
    FF=R(I)
    DO K=I,N
        BB=R(K)
        IF(K .EQ. 1) THEN
            CALL ROMBERG(DG, CS, AAA, ANTG)
        ELSE
            AA1=UL(K)-T0

```

```

        CALL ROMBERG(DG1,CS,AAA,ANTG)
        ENDIF
        B2(K,I)=ANTG
    ENDDO
    CS=UL(I)-T0
    CC=R(I)
    TT=UL(I)-T0
ENDDO
PRINT*, 'B2', ((B2(K,I), I=1,3), K=1,3)
C
C    FINDING VECTOR B3
C
    DO K=1,N
        B3(K)=0.0D0
    ENDDO
    DO K=1,N
        B3(K)=(R(K)-R(K-1))/DT
    ENDDO
    PRINT *, 'B3', B3(1), B3(2), B3(3)
C
C    FINDING MATRIX C0
C
    DO I=1,N
        DO J=1,N
            C0(I,J)=0.0D0
        ENDDO
    ENDDO
C
    T0=0.0D0
    DO J=1,N
        F=R(J)
        E=UL(J)
        DO I=1,J
            IC=I
            CALL ROMBERG(FUN,T0,E,COANT)
            C0(J,I)=COANT
        ENDDO
    ENDDO
    PRINT*, 'C0', ((A(I,J), J=1,3), I=1,3)
C
C    MULTIPLY MATRIX C0 BY THE COEFFICIENTS
C
    DO I=1,N
        DO J=1,N
            A(J,I)=0.0D0
        ENDDO
    ENDDO
    DO I=1,N
        DO J=1,N
            A(J,I)=C0(J,I)*COEF
        ENDDO
    ENDDO
    PRINT*, 'A', ((A(I,J), J=1,3), I=1,3)
C    MULTIPLY MATRIX B2 BY VECTOR B3

```



```

DO I=1,N
  B23(I)=0.0D0
ENDDO
DO I=1,N
  DO J=1,N
    B23(I)=B23(I)+B2(I,J)*B3(J)
  ENDDO
ENDDO
PRINT*, 'B23', B23(1), B23(2)

```

C
C
C

FINDING VECTOR B

```

DO I=1,N
  B(I)=0.0D0
ENDDO
DO I=1,N
  B(I)=B1(I)+B23(I)
ENDDO
DO I=1,N
  X(I)=B(I)
  DO J=1,N
    AA(I,J)=A(I,J)
  ENDDO
ENDDO
CALL LUDCMP(AA,N,NP,INDX,D)
CALL LUBKSB(AA,N,NP,INDX,X)
IDUM=-13
DO I=1,N
  X(I)=X(I)*(1.0+0.2*RAN3(IDUM))
ENDDO
CALL MPROVE(A,AA,N,NP,INDX,B,X)
ENDIF
WRITE(16,*) ' AN'
DO I=1,N
  WRITE(16,*) X(I)
ENDDO
DO I=1,20
  PP=X(N)
  DO J=N-1,1,-1
    PP=PP*I+X(J)
  ENDDO
  GG(I)=PP
  PRINT*,GG(I)
ENDDO

```

C

```

DT=1
DO I=1,NN
  READ(8,*)UL(I),R(I)
  PRINT*,R(I)
ENDDO
DO I=1,NN
  DO J=1,NN
    A(I,J)=0.0D0
  ENDDO

```

```

ENDDO
IC=1.0D0
CSS=0.0D0
DO I=1,NN
  AAAA=UL(I)
  DO K=I,NN
    F=0.0D0
    E=UL(K)
    CALL ROMBERG(FUN,CSS,AAAA,COANT)
    A(K,I)=COANT
  ENDDO
  CSS=UL(I)
ENDDO

```

C

```

DO I=1,NN
  DO J=1,NN
    B2(I,J)=0.0D0
  ENDDO
ENDDO

```

C

```

CC=R0
TT=T0
CS=0.0D0
DO I=1,NN
  AAA=UL(I)-T0
  FF=R(I)
  DO K=I,NN
    BB=0.0D0
    IF(K.EQ.1)THEN
      CALL ROMBERG(DG,CS,AAA,ANTG)
    ELSE
      AA1=UL(K)-T0
      CALL ROMBERG(DG1,CS,AAA,ANTG)
    ENDIF
    B2(K,I)=ANTG
  ENDDO
  CS=UL(I)-T0
  CC=R(I)
  TT=UL(I)-T0
ENDDO

```

C

```

DO K=1,NN
  B3(K)=0.0D0
ENDDO
DO K=1,NN
  B3(K)=(R(K)-R(K-1))/DT
ENDDO

```

C

```

DO I=1,NN
  B23(I)=0.0D0
ENDDO
DO I=1,NN
  DO J=1,NN
    B23(I)=B23(I)+B2(I,J)*B3(J)
  ENDDO
ENDDO

```

```

        ENDDO
ENDDO
C
DO I=1,NN
    A23(I)=0.0D0
ENDDO
C
DO I=1,NN
    DO J=1,NN
        A23(I)=A23(I)+A(I,J)*GG(J)
    ENDDO
ENDDO
C
DO I=1,NN
    TEM(I)=0.0D0
ENDDO
DO I=1,NN
    TEM(I)=1/CK*DSQRT(ALPHA/PI)*A23(I)-CL/C*B23(I)
    PRINT*,TEM(I)+TI
ENDDO
C
END
C*****C
C    SUBPROGRAM FUNCTION                                C
C*****C
FUNCTION FUN(T)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
COMMON/AL/ALPHA
COMMON/ICC/IC
COMMON/EE/E
COMMON/FF/F
IF(E.EQ.T)THEN
    FUN=0.0D0
ELSE
    FUN=T** (IC-1)/DSQRT(E-T)*DEXP(-(F**2)/4.DO/ALPHA/(E-T))
ENDIF
END
C
C*****C
C    SUBPROGRAM FUNCTION                                C
C*****C
FUNCTION DG(T)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
COMMON/ADT/DT
COMMON/AL/ALPHA
COMMON/MON/AAA
COMMON/MIN/FF
COMMON/MONA/BB
COMMON/AAD/TO
DATA R0/0.0D0/
DATA PI/3.14159265359D0/
IF(AAA.EQ.T) THEN
    DG=0.0D0
ELSE

```

```

      DG=1.DO/DSQRT(4.DO*PI*ALPHA*(AAA-T))*
&      (DEXP(-(BB-FF/DT*T)**2/(4.DO*ALPHA*(AAA-T)))+
&      DEXP(-(BB+FF/DT*T)**2/(4.DO*ALPHA*(AAA-T))))
    ENDIF
  END
C
C*****C
C      SUBPROGRAM FUNCTIONC
C*****C
      FUNCTION DG1(T)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DATA PI/3.14159265359D0/
      COMMON/ADT/DT
      COMMON/AL/ALPHA
      COMMON/MON/AAA
      COMMON/MONA/BB
      COMMON/AAD/TO
      COMMON/SARA/DD
      COMMON/MINA/CC
      COMMON/MIN/FF
      COMMON/SAR/TT
      COMMON/MON1/AA1
      IF(AA1.EQ.T)THEN
        DG1=0.0D0
      ELSE
        DG1=1.DO/DSQRT(4*PI*ALPHA*(AA1-T))*
&      (DEXP(-(BB-(CC+((FF-CC)/DT)*(T-TT))**2)/(4.DO*ALPHA*(AA1-T)))+
&      DEXP(-(BB+(CC+((FF-CC)/DT)*(T-TT))**2)/(4.DO*ALPHA*(AA1-T))))
      ENDIF
    END
C
C*****C
C      SUBPROGRAMC
C      ROMBERG INTEGRATIONC
C*****C
      SUBROUTINE ROMBERG(FUNC,A,B,RESULT)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      EXTERNAL FUNC
      PARAMETER(MAX=40,EPS=0.0001D0)
      DIMENSION T(MAX,MAX)
C
      T(1,1)=(B-A)*(FUNC(A)+FUNC(B))/2.0D0
      T(1,2)=T(1,1)/2.0D0+(B-A)*FUNC((A+B)/2.0D0)/2.0D0
      T(2,1)=(4.0D0*T(1,2)-T(1,1))/3.0D0
      J=3
C
C-----SUCCESSIVE APPLICATION OF TRAPEZOIDAL RULE
C
50    DELX=(B-A)/2.0D0**(J-1)
      X=A-DELX
      N=2**(J-2)
      SUM=0.0D0
      DO 100 I=1,N
        X=X+2.0D0*DELX

```

```

        SUM=SUM+FUNC(X)
100  CONTINUE
        T(1,J)=T(1,J-1)/2.0D0+DELX*SUM
C
C-----EXTRAPOLATION
C
        DO 200 L=2,J
            K=J+1-L
            T(L,K)=(4.0D0**(L-1)*T(L-1,K+1)-T(L-1,K))/
$          (4.0D0**(L-1)-1.0D0)
200  CONTINUE
C
C-----CHECK ACCURACY CRITERION
C
        IF(T(J,1) .EQ. 0.0D0) THEN
            RESULT=T(J,1)
            GO TO 111
        END IF
C
        IF(DABS((T(J,1)-T(J-1,1))/T(J,1)) .GE. EPS) THEN
            J=J+1
            IF(J.GT.MAX) THEN
                PAUSE 'TOO MANY STEPS'
                GO TO 111
            ELSE
                GO TO 50
            END IF
        ELSE
            RESULT=T(J,1)
        END IF
111  RETURN
        END
C*****C
C  SUBROUTINE MPROVE                                     C
C*****C
        SUBROUTINE MPROVE(A,ALUD,N,NP,INDX,B,X)
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        PARAMETER (NMAX=100)
        DIMENSION A(NP,NP),ALUD(NP,NP),INDX(N),B(N),X(N),R(NMAX)
        REAL*8 SDP
        DO 12 I=1,N
            SDP=-B(I)
            DO 11 J=1,N
                SDP=SDP+DBLE(A(I,J))*DBLE(X(J))
11      CONTINUE
            R(I)=SDP
12      CONTINUE
        CALL LUBKSB(ALUD,N,NP,INDX,R)
        DO 13 I=1,N
            X(I)=X(I)-R(I)
13      CONTINUE
        RETURN
        END

```

```

C*****C
C  SUBROUTINE LU DECOMPOSITIONC
C*****C
SUBROUTINE LUDCMP(A,N,NP,INDX,D)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
PARAMETER (NMAX=100,TINY=1.0E-20)
DIMENSION A(NP,NP),INDX(N),VV(NMAX)
D=1.0D0
DO 12 I=1,N
  AAMAX=0.0D0
  DO 11 J=1,N
    IF (ABS(A(I,J)).GT.AAMAX) AAMAX=ABS(A(I,J))
11  CONTINUE
    IF (AAMAX.EQ.0.) PAUSE 'SINGULAR MATRIX.'
    VV(I)=1.0D0/AAMAX
12  CONTINUE
    DO 19 J=1,N
      IF (J.GT.1) THEN
        DO 14 I=1,J-1
          SUM=A(I,J)
          IF (I.GT.1) THEN
            DO 13 K=1,I-1
              SUM=SUM-A(I,K)*A(K,J)
13          CONTINUE
              A(I,J)=SUM
            ENDIF
14          CONTINUE
          ENDIF
          AAMAX=0.0D0
          DO 16 I=J,N
            SUM=A(I,J)
            IF (J.GT.1) THEN
              DO 15 K=1,J-1
                SUM=SUM-A(I,K)*A(K,J)
15          CONTINUE
                A(I,J)=SUM
              ENDIF
              DUM=VV(I)*ABS(SUM)
              IF (DUM.GE.AAMAX) THEN
                IMAX=I
                AAMAX=DUM
              ENDIF
16          CONTINUE
            IF (J.NE.IMAX) THEN
              DO 17 K=1,N
                DUM=A(IMAX,K)
                A(IMAX,K)=A(J,K)
                A(J,K)=DUM
17          CONTINUE
              D=-D
              VV(IMAX)=VV(J)
            ENDIF
            INDX(J)=IMAX
            IF (J.NE.N) THEN

```

```

        IF(A(J,J).EQ.0.)A(J,J)=TINY
        DUM=1./A(J,J)
        DO 18 I=J+1,N
            A(I,J)=A(I,J)*DUM
18      CONTINUE
        ENDIF
19     CONTINUE
        IF(A(N,N).EQ.0.)A(N,N)=TINY
        RETURN
        END

C*****C
C  SUBROUTINE BACKSUBSTITUTION                                     C
C*****C
        SUBROUTINE LUBKSB(A,N,NP,INDX,B)
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        DIMENSION A(NP,NP),INDX(N),B(N)
        II=0
        DO 12 I=1,N
            LL=INDX(I)
            SUM=B(LL)
            B(LL)=B(I)
            IF (II.NE.0)THEN
                DO 11 J=II,I-1
                    SUM=SUM-A(I,J)*B(J)
11      CONTINUE
            ELSE IF (SUM.NE.0.) THEN
                II=I
            ENDIF
            B(I)=SUM
12     CONTINUE
        DO 14 I=N,1,-1
            SUM=B(I)
            IF(I.LT.N)THEN
                DO 13 J=I+1,N
                    SUM=SUM-A(I,J)*B(J)
13      CONTINUE
            ENDIF
            B(I)=SUM/A(I,I)
14     CONTINUE
        RETURN
        END

C*****C
C  SUBROUTINE RANDOM                                             C
C*****C
        FUNCTION RAN3(IDUM)
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        PARAMETER (MBIG=1000000000,MSEED=161803398,MZ=0,FAC=1.E-9)
        DIMENSION MA(55)
        DATA IFF /0/
        IF(IDUM.LT.0.OR.IFF.EQ.0)THEN
            IFF=1
            MJ=MSEED-IABS(IDUM)
            MJ=MOD(MJ,MBIG)

```



```

      MA(55)=MJ
      MK=1
      DO 11 I=1,54
        II=MOD(21*I,55)
        MA(II)=MK
        MK=MJ-MK
        IF(MK.LT.MZ)MK=MK+MBIG
        MJ=MA(II)
11      CONTINUE
      DO 13 K=1,4
        DO 12 I=1,55
          MA(I)=MA(I)-MA(1+MOD(I+30,55))
          IF(MA(I).LT.MZ)MA(I)=MA(I)+MBIG
12      CONTINUE
13      CONTINUE
      INEXT=0
      INEXTP=31
      IDUM=1
ENDIF
INEXT=INEXT+1
IF(INEXT.EQ.56)INEXT=1
INEXTP=INEXTP+1
IF(INEXTP.EQ.56)INEXTP=1
MJ=MA(INEXT)-MA(INEXTP)
IF(MJ.LT.MZ)MJ=MJ+MBIG
MA(INEXT)=MJ
RAN3=MJ*FAC
RETURN
END

```

APPENDIX E
SSM FORTRAN PROGRAM FOR INVERSE STEFAN PROBLEM

```

C*****C
C      SOLUTION OF 1D INVERSE STEFAN PROBLEMS IN A SEMI-INFINITE      C
C      MEDIUM WITH OR WITHOUT SUBCOOLING BY THE SOURCE-AND-SINK      C
C      METHOD USING INCREMENTAL APPROACH.  BOUNDARY CONDITION          C
C      FOUND VIA ASSUMED FLUX CONDITION                                C
C                                                                      C
C      INPUT FILE UNIT NUMBER IS SET AT 12                            C
C      OUTPUT FILE UNIT NUMBER IS SET AT 16                            C
C      NOTATIONS                                                        C
C                                                                      C
C      TO      =TIME WHEN PHASE CHANGE STARTS                          C
C      DT      =TIME-STEP SIZE                                          C
C      N       =NUMBER OF ENTRIES IN INPUT FILE                        C
C      ALPHA   =THERMAL DIFFUSIVITY                                    C
C      CK      =THERMAL CONDUCTIVITY                                   C
C      C       =SPECIFIC HEAT                                          C
C      CL      =LATENT HEAT OF FUSION                                  C
C      RHO     =DENSITY                                                 C
C      TI      =INITIAL TEMPERATURE                                    C
C      TM      =MELTING TEMPERATURE                                   C
C*****C
C
C-----MAIN PROGRAM
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER (NP=250)
      DATA PI/3.14159265359D0/
      DATA RO/0.0D0/
      DIMENSION R(NP),UL(NP),CO(NP,NP),A(NP,NP)
      DIMENSION B3(NP),B23(NP),B2(NP,NP),X(NP)
      DIMENSION GG(NP),GLL(NP),A23(NP),TEM(NP)
      CHARACTER *40 FNAME
      CHARACTER*12 MQ
      COMMON/MINA/CC
      COMMON/MIN/FF
      COMMON/AL/ALPHA
      COMMON/ADT/DT
      COMMON/AAD/TO
      COMMON/MON/AAA
      COMMON/MONA/BB
      COMMON/EE/E
      COMMON/FF/F
      COMMON/ICC/IC
      COMMON/SAR/TT
      COMMON/SARA/DD

```

```
COMMON/MON1/AA1
INTEGER RDSTAT,OPSTAT
EXTERNAL FUN,FUN1, DG, DG1
```

```
10 CONTINUE
PRINT*, 'ENTER FILE NAME OR MQ TO QUIT: '
READ '(A)', FNAME
OPEN(UNIT=1, FILE=FNAME, STATUS='OLD', IOSTAT=OPSTAT)
IF(.NOT.(OPSTAT .EQ. 0 .OR. FNAME .EQ. 'MQ')) GO TO 10
IF (FNAME .NE. 'MQ') THEN
```

```
C
PRINT*, 'DT, N, TO, TM, TI: '
READ*, DT, N, TO, TM, TI
READ(12,*)C, CL, CK, RHO
ALPHA=CK/RHO/C
COEF=CL*CK/C*DSQRT(PI/ALPHA)
COEF1=CK*TM*DSQRT(PI/ALPHA)
```

```
C
C READ DATA FILE
```

```
C
DO I=1,N
  READ(UNIT=1, FMT=*, IOSTAT=RDSTAT) UL(I), R(I)
C  PRINT*, UL(I), R(I)
ENDDO
```

```
C
C FINDING MATRIX B2
```

```
C
DO K=1,N
  DO I=1,N
    B2(I,K)=0.0DO
  ENDDO
ENDDO
```

```
C
CC=RO
TT=TO
CS=0.0DO
DO I=1,N
  AAA=UL(I)-TO
  FF=R(I)
  DO K=I,N
    BB=R(K)
    IF(K .EQ. 1) THEN
      CALL ROMBERG(DG, CS, AAA, ANTG)
    ELSE
      AA1=UL(K)-TO
      CALL ROMBERG(DG1, CS, AAA, ANTG)
    ENDIF
    B2(K,I)=ANTG
  ENDDO
  CS=UL(I)-TO
  CC=R(I)
  TT=UL(I)-TO
ENDDO
PRINT*, 'B2', ((B2(K,I), I=1,3), K=1,3)
```

```

C
c  FINDING VECTOR B3
c
DO K=1,N
  B3(K)=0.0D0
ENDDO
DO K=1,N
  B3(K)=(R(K)-R(K-1))/DT
ENDDO
PRINT *, 'B3', B3(1), B3(2), B3(3)

C
C  FINDING MATRIX A
c
DO I=1,N
  DO J=1,N
    A(I,J)=0.0D0
  ENDDO
ENDDO

C
CSS=0.0D0
DO I=1,N
  AAAA=UL(I)
  DO K=I,N
    F=R(K)
    E=UL(K)
    CALL ROMBERG(FUN,CSS,AAAA,COANT)
    A(K,I)=COANT
  ENDDO
  CSS=UL(I)
ENDDO
PRINT*, 'A', ((A(I,J), J=1,3), I=1,3)

C
c  MULTIPLY MATRIX B2 BY VECTOR B3
c
DO I=1,N
  B23(I)=0.0D0
ENDDO
DO I=1,N
  DO J=1,N
    B23(I)=B23(I)+B2(I,J)*B3(J)
  ENDDO
ENDDO
PRINT*, 'B23', B23(1), B23(2)

C
GG(1)=(COEF1+COEF*B23(1))/A(1,1)
XX=0.0D0
DO I=2,N
  DO K=1,I-1
    GLL(I)=GG(K)*A(I,K)+XX
    XX=GLL(I)
  ENDDO
  GG(I)=(COEF1+COEF*B23(I)-GLL(I))/A(I,I)
  XX=0.0D0
ENDDO

```

```

WRITE(11,*)'  G(I)'
DO I=1,N
  WRITE(11,*) GG(I)
ENDDO

```

C
C
C

```

FINDING TEMPERATURE

```

```

DT=1
DO I=1,N
  DO J=1,N
    A(I,J)=0.0D0
  ENDDO
ENDDO
CSS=0.0D0
DO I=1,N
  AAAA=UL(I)
  DO K=I,N
    F=0.0D0
    E=UL(K)
    CALL ROMBERG(FUN,CSS,AAAA,COANT)
    A(K,I)=COANT
  ENDDO
  CSS=UL(I)
ENDDO

```

C

```

DO I=1,N
  DO J=1,N
    B2(I,J)=0.0D0
  ENDDO
ENDDO
CC=R0
TT=T0
CS=0.0D0
DO I=1,N
  AAA=UL(I)-T0
  FF=R(I)
  DO K=I,N
    BB=0.0D0
    IF(K.EQ.1)THEN
      CALL ROMBERG(DG,CS,AAA,ANTG)
    ELSE
      AA1=UL(K)-T0
      CALL ROMBERG(DG1,CS,AAA,ANTG)
    ENDIF
    B2(K,I)=ANTG
  ENDDO
  CS=UL(I)-T0
  CC=R(I)
  TT=UL(I)-T0
ENDDO

```

C

```

DO K=1,N
  B3(K)=0.0D0
ENDDO

```

```

DO K=1,N
  B3(K)=(R(K)-R(K-1))/DT
ENDDO
C
DO I=1,N
  B23(I)=0.0D0
ENDDO
DO I=1,N
  DO J=1,N
    B23(I)=B23(I)+B2(I,J)*B3(J)
  ENDDO
ENDDO
C
DO I=1,N
  A23(I)=0.0D0
ENDDO
DO I=1,N
  DO J=1,N
    A23(I)=A23(I)+A(I,J)*GG(J)
  ENDDO
ENDDO
C
DO I=1,N
  TEM(I)=0.0D0
ENDDO
DO I=1,N
  TEM(I)=1/CK*DSQRT(ALPHA/PI)*A23(I)-CL/C*B23(I)
  PRINT(16,*)TEM(I)+TI
ENDDO
ENDIF
END
C
C*****C
C  SUBPROGRAM FUNCTION C
C*****C
FUNCTION FUN(T)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
COMMON/AL/ALPHA
COMMON/EE/E
COMMON/FF/F
COMMON/ICC/IC
IF(E.EQ.T)THEN
  FUN=0.0D0
ELSE
  FUN=1.0D0/DSQRT(E-T)*DEXP(-(F**2)/4.DO/ALPHA/(E-T))
ENDIF
END
C
C*****C
C  SUBPROGRAM FUNCTION C
C*****C
FUNCTION DG(T)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
COMMON/ADT/DT

```

```

COMMON/AL/ALPHA
COMMON/MON/AAA
COMMON/MIN/FF
COMMON/MONA/BB
COMMON/AAD/TO
DATA RO/0.DO/
DATA PI/3.14159265359D0/
IF(AAA.EQ.T) THEN
    DG=0.0D0
ELSE
    DG=1.DO/DSQRT(4.DO*PI*ALPHA*(AAA-T))*
&      (DEXP(-(BB-FF/DT*T)**2/(4.DO*ALPHA*(AAA-T)))+
&      DEXP(-(BB+FF/DT*T)**2/(4.DO*ALPHA*(AAA-T))))
ENDIF
END

C
C*****C
C      SUBPROGRAM FUNCTION                                     C
C*****C
FUNCTION DG1(T)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
DATA PI/3.14159265359D0/
COMMON/ADT/DT
COMMON/AL/ALPHA
COMMON/MON/AAA
COMMON/MONA/BB
COMMON/AAD/TO
COMMON/SARA/DD
COMMON/MINA/CC
COMMON/MIN/FF
COMMON/SAR/TT
COMMON/MON1/AA1
IF(AA1.EQ.T)THEN
    DG1=0.0D0
ELSE
    DG1=1.DO/DSQRT(4*PI*ALPHA*(AA1-T))*
&      (DEXP(-(BB-(CC+((FF-CC)/DT)*(T-TT)))**2)/(4.DO*ALPHA*(AA1-T)))
&      +DEXP(-(BB+(CC+((FF-CC)/DT)*(T-TT)))**2)/(4.DO*ALPHA*(AA1-T)))
ENDIF
END

C
C*****C
C      SUBROUTINE                                             C
C      ROMBERG INTEGRATION                                    C
C*****C
SUBROUTINE ROMBERG(FUNC,A,B,RESULT)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
EXTERNAL FUNC
PARAMETER(MAX=50,EPS=0.0001D0)
DIMENSION T(MAX,MAX)

C
T(1,1)=(B-A)*(FUNC(A)+FUNC(B))/2.0D0
T(1,2)=T(1,1)/2.0D0+(B-A)*FUNC((A+B)/2.0D0)/2.0D0
T(2,1)=(4.0D0*T(1,2)-T(1,1))/3.0D0

```



```

      J=3
C
C-----SUCCESSIVE APPLICATION OF TRAPEZOIDAL RULE
C
50   DELX=(B-A)/2.0D0**(J-1)
      X=A-DELX
      N=2**(J-2)
      SUM=0.0D0
      DO 100 I=1,N
          X=X+2.0D0*DELX
          SUM=SUM+FUNC(X)
100  CONTINUE
      T(1,J)=T(1,J-1)/2.0D0+DELX*SUM
C
C-----EXTRAPOLATION
C
      DO 200 L=2,J
          K=J+1-L
          T(L,K)=(4.0D0**(L-1)*T(L-1,K+1)-T(L-1,K))/
$      (4.0D0**(L-1)-1.0D0)
200  CONTINUE
C
C-----CHECK ACCURACY CRITERION
C
      IF(T(J,1) .EQ. 0.0D0) THEN
          RESULT=T(J,1)
          GO TO 111
      END IF
C
      IF(DABS((T(J,1)-T(J-1,1))/T(J,1)) .GE. EPS) THEN
          J=J+1
          IF(J.GT.MAX) THEN
              PAUSE 'TOO MANY STEPS'
              GO TO 111
          ELSE
              GO TO 50
          END IF
      ELSE
          RESULT=T(J,1)
      END IF
111  RETURN
      END

```

APPENDIX F
SSM FORTRAN PROGRAM FOR ABLATION PROBLEM

```

C*****C
C      SOLUTION OF 1D ABLATION PROBLEM WITH ONE MOVING BOUNDARY      C
C      IN A SEMI-INFINITE MEDIUM WITH OR WITHOUT SUBCOOLING          C
C      BY THE SOURCE-AND-SINK METHOD                                   C
C                                                                      C
C      INPUT FILE UNIT NUMBER IS SET AT 13                            C
C      OUTPUT FILE UNIT IS SET AT 8 AND 14                            C
C                                                                      C
C              NOTATIONS                                             C
C                                                                      C
C      TMI      =TIME WHEN PHASE CHANGE STARTS                       C
C      DELTM    =TIME-STEP SIZE                                     C
C      ALPHA    =THERMAL DIFFUSIVITY                               C
C      CK       =THERMAL CONDUCTIVITY                              C
C      CP       =SPECIFIC HEAT                                     C
C      CL       =LATENT HEAT OF FUSION                             C
C      R        =ABLATED SURFACE POSITION                           C
C      RV       =SURFACE VELOCITY                                  C
C      RHO      =DENSITY                                           C
C      TM       =PHASE CHANGE TEMPERATURE                          C
C*****C
C
C-----MAIN PROGRAM
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER (MAXF=2000,NP=15)
      DATA PI/3.14159265359D0/
      COMMON/C/ALPHA
      DIMENSION XX(NP)
      COMMON/T/NF,INF,TMI,TMF,DELT
      COMMON/C1/CK,RHO,CP,CL,TM,U
      COMMON/R1/R(MAXF),RV(MAXF)
      COMMON/XXF/XX(NP)
      COMMON/GH/NK
      COMMON/GG/G(MAXF)
      COMMON/SARA/NNN
      COMMON/EPSILON/EE
      COMMON/Q1/Q(MAXF)
      COMMON/GAS/SUMA,SUMB,SUME,SAA,SDD,QINTT,QINT
      EXTERNAL TFUNC1,TFUNC2
C
      READ(13,*)CK,RHO,CP,CL,TM
      NTRIAL=1000
      WRITE(*,*)'TMI,DELT='
      READ(*,*) TMI,DELT

```

```

WRITE(*,*)'TOTAL NO. OF ITERATION TIME STEP MF=?'
READ(*,*)MF
WRITE(14,*)'*****'
WRITE(14,*)'INPUT DATA'
WRITE(14,*)' '
WRITE(14,*)'MELTING TEMPERATURE=',TM , 'C'
WRITE(14,*)'THERMAL CONDUCTIVITY=',CK , 'KJ/S.CM.C'
WRITE(14,*)'DENSITY=',RHO , 'G/CUBIC CM'
WRITE(14,*)'SPECIFIC HEAT CAPACITY=',CP , 'KJ/G.C'
WRITE(14,*)'LATENT HEAT=',CL , 'KJ/G'
WRITE(14,*)' '
WRITE(14,*)'*****'
WRITE(14,200)
200  FORMAT(8X,'TIME',17X,'Q(T)',17X,'R(T)',14X,'DR/DT')
WRITE(14,*)' '
C
C-----INITIALIZE R,RV,F
C
DO I=1, MAXF
  R(I)=0.0D0
  RV(I)=0.0D0
  Q(I)=0.0D0
ENDDO
C
ALPHA=CK/RHO/CP
C
DO NF=1, MF
  XNF=NF
  TFF=TMI+XNF*DELTM
C
C-----ASSUME INTERFACE POSITION INTERVAL R(NF)
C
TOL=1.D-20
IF(NF .EQ. 1) THEN
  WRITE(*,*)TFF,' ENTER INITIAL R1,R2'
  READ(*,*)R1,R2
ELSE
  R1=R(NF-1)
  R2=5.D0*R1
ENDIF
EE=0.0D0
R(NF)=ZBRENT(TFUNC1,R1,R2,TOL)
Q(NF)=(TM-SUMA+CL/CP*SUMB-QINT)/SAA
WRITE(14,1000)TFF,Q(NF),R(NF),R(NF)/TFF
1000  FORMAT(3X,F9.4,5X,F23.9,5X,E14.7,5X,E14.7)
ENDDO
END
C*****C
C  SUBPROGRAM FUNCTION C
C*****C
FUNCTION TFUNC1(X)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
PARAMETER (MAXF=2000,PI=3.141592653589793D0,NP=2)
COMMON/C/ALPHA

```

```

COMMON/EPSILON/EE
COMMON/T/NF, INF, TMI, TMF, DELTM
COMMON/C1/CK, RHO, CP, CL, TM, U
COMMON/R1/R(MAXF), RV(MAXF)
COMMON/Q1/Q(MAXF)
COMMON/GH/NK
COMMON/GG/G(MAXF)
COMMON/GAS/SUMA, SUMB, SUMD, SUME, SAA, SDD, QINTT, QINT
EXTERNAL FUNC1, FUNC2A, FUNC2B, FUNC3, FUNC33, FUNC22A
EXTERNAL FUNC11, FUNC22B
EXTERNAL FUNC4, FUNC5A, FUNC5B, FUNC6, FUNC44, FUNC66
EXTERNAL FUNC55A1, FUNC55B, FUNC55A2, FUNC5A1, FUNC5A2
R(NF)=X
EE=0.0D0

```

C

```

TFF2=TMI+NF*DELT
TFF=TFF2
TFF1=TMI+(NF-1)*DELT
IF(NF .EQ. 1) THEN
    RV(NF)=R(NF)/(TFF2-TFF1)
ELSE
    RV(NF)=(R(NF)-R(NF-1))/(TFF2-TFF1)
ENDIF

```

C

C-----COMPUTE PART A

C

```

SUMA=0.0D0
DO INF=1, NF
    TF2=TMI+INF*DELT
    TF1=TMI+(INF-1)*DELT
    IF(INF .EQ. NF) THEN
        AA=DSQRT(TFF2-TF1)
        BB=DSQRT(TFF2-TF2)
        CALL ROMBERG(FUNC1, AA, BB, SAA)
    ELSE
        AA=DSQRT(TFF2-TF1)
        BB=DSQRT(TFF2-TF2)
        CALL ROMBERG(FUNC1, AA, BB, A)
        A=Q(INF)*A*DSQRT(ALPHA/PI)/CK
        SUMA=SUMA+A
    ENDIF
ENDDO
SAA=DSQRT(ALPHA/PI)/CK*SAA

```

C

C-----COMPUTE PART B

C

```

SUMB1=0.0D0
SUMB2=0.0D0
DO INF=1, NF
    TF2=TMI+INF*DELT
    TF1=TMI+(INF-1)*DELT
    AA=DSQRT(TFF2-TF1)
    BB=DSQRT(TFF2-TF2)
    CALL ROMBERG(FUNC2A, AA, BB, BB1)

```

```

      CALL ROMBERG(FUNC2B,AA,BB,BB2)
      BB1=RV(INF)*BB1
      BB2=RV(INF)*BB2
      SUMB1=SUMB1+BB1
      SUMB2=SUMB2+BB2
    ENDDO
    SUMB=SUMB1+SUMB2
C
C-----COMPUTE PART C
C
    IF(TMI .EQ. 0.0D0) THEN
      QINT=0.0D0
    ELSE
      CC1=DSQRT(TFF)
      CC2=DSQRT(TFF-TMI)
      CALL ROMBERG(FUNC3,CC1,CC2,QINT)
      QINT=QINT*DSQRT(ALPHA/PI)/CK
    ENDIF
C
C-----COMPUTE PART D
C
    SUMD=0.0D0
    DO INF=1,NF-1
      TF2=TMI+INF*DELTM
      TF1=TMI+(INF-1)*DELTM
      A1=4.0D0*DSQRT(ALPHA)/(R(NF)+EE)
      A2=(R(NF)+EE)/DSQRT(4.0D0*ALPHA*(TFF2-TF1))
      A3=(R(NF)+EE)/DSQRT(4.0D0*ALPHA*(TFF2-TF2))
      CALL ROMBERG(FUNC5A,A2,A3,D)
      D=A1*D
      D=Q(INF)*D*R(NF)/2.0D0/DSQRT(ALPHA*PI)
      SUMD=SUMD+D
    ENDDO
    A2=(R(NF)+EE)/DSQRT(4.0D0*ALPHA*(TFF2-TFF1))
    SDD=ERFC(A2)
C
C-----COMPUTE PART E
C
    SUME1=0.0D0
    SUME2=0.0D0
    DO INF=1,NF
      TF2=TMI+INF*DELTM
      TF1=TMI+(INF-1)*DELTM
      IF(INF .EQ. 1)THEN
        AM=RV(INF)
        BN=-R(INF)*TF1/(TF2-TF1)
      ELSE
        AM=RV(INF)
        BN=(R(INF-1)*TF2-R(INF)*TF1)/(TF2-TF1)
      ENDIF
      IF(INF .EQ. NF)THEN
        C1=(R(NF)+EE)-2.0D0*AM*TFF2+AM*TF1-BN
        C2=DSQRT(4.0D0*ALPHA*(TFF2-TF1))
        CC=C1/C2
      
```

```

C01=1.0D0/2.0D0*DEXP(-AM*((R(NF)+EE)-AM*TFF2-BN)/ALPHA)
E1=C01*ERFC(CC)
F1=((R(NF)+EE)+2.0D0*AM*TFF2-AM*TF1+BN)
FF=F1/C2
C02=1.0D0/2.0D0*DEXP(AM*((R(NF)+EE)+AM*TFF2+BN)/ALPHA)
E2=C02*ERFC(FF)
ELSE
  C01=1.0D0/DSQRT(PI)*DEXP(-AM*((R(NF)+EE)-AM*TFF2-BN)/ALPHA)
  C02=1.0D0/DSQRT(PI)*DEXP(AM*((R(NF)+EE)+AM*TFF2+BN)/ALPHA)
  C1=(R(NF)+EE)-2.0D0*AM*TFF2+AM*TF1-BN
  C2=DSQRT(4.0D0*ALPHA*(TFF2-TF1))
  CC=C1/C2
  C3=(R(NF)+EE)-2.0D0*AM*TFF2+AM*TF2-BN
  C4=DSQRT(4.0D0*ALPHA*(TFF2-TF2))
  CC1=C3/C4
  CALL ROMBERG(FUNC5B,CC,CC1,E1)
  E1=C01*E1
  F1=((R(NF)+EE)+2.0D0*AM*TFF2-AM*TF1+BN)
  FF=F1/C2
  F2=((R(NF)+EE)+2.0D0*AM*TFF2-AM*TF2+BN)
  FF1=F2/C4
  CALL ROMBERG(FUNC5B,FF,FF1,E2)
  E2=C02*E2
ENDIF
E1=RV(INF)*E1
E2=RV(INF)*E2
SUME1=SUME1+E1
SUME2=SUME2+E2
ENDDO
SUME=SUME1+SUME2
C
C-----COMPUTE TO TERM
C
  IF(TMI.EQ.0.0D0) THEN
    QINTT=0.0D0
  ELSE
    CC1=1.0D0/DSQRT(TFF)
    CC2=1.0D0/DSQRT(TFF-TMI)
    CALL ROMBERG(FUNC6,CC1,CC2,QINTT)
    QINTT=QINTT*R(NF)/2.0D0/DSQRT(ALPHA*PI)
  ENDIF
  Q2=GX(TFF2)-QINTT+RHO*CL*SUME-RHO*CL*RV(NF)-SUMD
  Q3=TM-QINT+CL/CP*SUMB-SUMA
  TFUNC1=SDD*Q3-SAA*Q2
  RETURN
END
C*****C
C  SUBPROGRAM FUNCTION C
C  FOR TFUNC1 C
C*****C
  FUNCTION FUNC1(X)
  IMPLICIT DOUBLE PRECISION(A-H,O-Z)
  PARAMETER (MAXF=2000)

```



```

COMMON/C/ALPHA
DATA PI/3.14159265359D0/
COMMON/C1/CK,RHO,CP,CL,TM,U
COMMON/T/NF,INF,TMI,TMF,DELTM
COMMON/R1/R(MAXF),RV(MAXF)

C
TFF=TMI+NF*DELTM
IF(X .EQ. 0.0D0) THEN
    FUNC1=0.0D0
ELSE
    ARG=-R(NF)**2/4.0D0/ALPHA/X/X
    FUNC1=-2.0D0*DEXP(ARG)
ENDIF
RETURN
END

C
C*****C
C SUBPROGRAM FUNCTION C
C*****C
FUNCTION FUNC2A(X)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
PARAMETER (MAXF=2000,PI=3.141592653589793D0)
COMMON/C/ALPHA
COMMON/T/NF,INF,TMI,TMF,DELTM
COMMON/R1/R(MAXF),RV(MAXF)

C
TFF=TMI+NF*DELTM
TF2=TMI+INF*DELTM
TF1=TMI+(INF-1)*DELTM
IF(INF .EQ. 1) THEN
    A=RV(INF)
    Z=-R(NF)*TF1/(TF2-TF1)
ELSE
    A=RV(INF)
    Z=(R(INF-1)*TF2-R(INF)*TF1)/(TF2-TF1)
END IF
RX=A*(TFF-X**2)+Z

C
IF(X .EQ. 0.0D0) THEN
    FUNC2A=-2.0D0/DSQRT(4*PI*ALPHA)
ELSE
    FUNC2A=-2.0D0/DSQRT(4*PI*ALPHA)
$    *DEXP(-(R(NF)-RX)**2/4.0D0/ALPHA/X/X)
ENDIF
RETURN
END

C
C*****C
C SUBPROGRAM FUNCTION C
C*****C
FUNCTION FUNC2B(X)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
PARAMETER (MAXF=2000,PI=3.141592653589793D0)
COMMON/C/ALPHA

```



```

COMMON/T/NF,INF,TMI,TMF,DELTM
COMMON/R1/R(MAXF),RV(MAXF)
C
TFF=TMI+NF*DELTM
TF2=TMI+INF*DELTM
TF1=TMI+(INF-1)*DELTM
IF(INF.EQ.1) THEN
  A=RV(INF)
  Z=-R(NF)*TF1/(TF2-TF1)
ELSE
  A=RV(INF)
  Z=(R(INF-1)*TF2-R(INF)*TF1)/(TF2-TF1)
END IF
RX=A*(TFF-X**2)+Z
C
IF(X.EQ.0.0D0) THEN
  FUNC2B=0.0D0
ELSE
  FUNC2B=-2.0D0/DSQRT(4*PI*ALPHA)
$      *DEXP(-(R(NF)+RX)**2/4.0D0/ALPHA/X/X)
END IF
RETURN
END
C
C*****C
C      SUBPROGRAM FUNCTIONC
C*****C
FUNCTION FUNC3(X)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
PARAMETER (MAXF=2000)
COMMON/C/ALPHA
DATA PI/3.14159265359D0/
COMMON/C1/CK,RHO,CP,CL,TM,U
COMMON/T/NF,INF,TMI,TMF,DELTM
COMMON/R1/R(MAXF),RV(MAXF)
C
TFF=TMI+NF*DELTM
FUNC3=-2.0D0*GX(TFF-X**2)*DEXP(-(R(NF)**2/4.0D0/ALPHA/X/X)
RETURN
END
C
C
C
C*****C
C      SUBPROGRAM FUNCTIONC
C*****C
FUNCTION FUNC4(X)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
PARAMETER (MAXF=2000)
COMMON/C/ALPHA
COMMON/T/NF,INF,TMI,TMF,DELTM
COMMON/R1/R(MAXF),RV(MAXF)
COMMON/EPSILON/EE
C

```

```

TFF=TMI+NF*DELTM
FUNC4=2.0D0*DEXP(-(R(NF)+EE)**2/4.0D0/ALPHA*X*X)
RETURN
END

```

```

C
C*****C
C      SUBPROGRAM FUNCTIONC
C*****C
      FUNCTION FUNC5A(X)
      IMPLICIT DOUBLE PRECISION(A-H,0-Z)
      PARAMETER (MAXF=2000)
      DATA PI/3.14159265359D0/
      COMMON/C/ALPHA
      COMMON/T/NF, INF, TMI, TMF, DELTM
      COMMON/R1/R(MAXF), RV(MAXF)
      COMMON/EPSILON/EE
      FUNC5A=DEXP(-X*X)
      RETURN
      END

```

```

C
C*****C
C      SUBPROGRAM FUNCTIONC
C*****C
      FUNCTION FUNC5B(X)
      IMPLICIT DOUBLE PRECISION(A-H,0-Z)
      PARAMETER (MAXF=2000)
      DATA PI/3.14159265359D0/
      COMMON/C/ALPHA
      COMMON/T/NF, INF, TMI, TMF, DELTM
      COMMON/R1/R(MAXF), RV(MAXF)
      COMMON/EPSILON/EE

```

```

C
      FUNC5B=DEXP(-X*X)
      RETURN
      END

```

```

C
C*****C
C      SUBPROGRAM FUNCTIONC
C*****C
      FUNCTION FUNC6(X)
      IMPLICIT DOUBLE PRECISION(A-H,0-Z)
      PARAMETER (MAXF=2000)
      COMMON/C/ALPHA
      COMMON/T/NF, INF, TMI, TMF, DELTM
      COMMON/R1/R(MAXF), RV(MAXF)
      COMMON/EPSILON/EE

```

```

C
      TFF=TMI+NF*DELTM
      FUNC6=2.0D0*GX(TFF-1.0D0/X/X)*DEXP(-(R(NF)+EE)**2/4.0D0/ALPHA*X*X)
      RETURN
      END

```

```

C

```

```

C*****C
C    SUBPROGRAM FUNCTION                                C
C*****C
    FUNCTION GX(X)
    IMPLICIT DOUBLE PRECISION(A-H,O-Z)
    GX=0.01D0
    END
C*****C
C    SUBPROGRAM                                C
C    ROMBERG INTEGRATION                        C
C*****C
    SUBROUTINE ROMBERG(FUNC,A,B,RESULT)
    IMPLICIT DOUBLE PRECISION(A-H,O-Z)
    EXTERNAL FUNC
    PARAMETER(MAX=50,EPS=0.0001D0)
    DIMENSION T(MAX,MAX)
C
    T(1,1)=(B-A)*(FUNC(A)+FUNC(B))/2.0D0
    T(1,2)=T(1,1)/2.0D0+(B-A)*FUNC((A+B)/2.0D0)/2.0D0
    T(2,1)=(4.0D0*T(1,2)-T(1,1))/3.0D0
    J=3
C
C-----SUCCESSIVE APPLICATION OF TRAPEZOIDAL RULE
C
100  DELX=(B-A)/2.0D0**(J-1)
      X=A-DELX
      N=2**(J-2)
      SUM=0.0D0
      DO 100 I=1,N
          X=X+2.0D0*DELX
          SUM=SUM+FUNC(X)
100  CONTINUE
      T(1,J)=T(1,J-1)/2.0D0+DELX*SUM
C
C-----EXTRAPOLATION
C
      DO 200 L=2,J
          K=J+1-L
          T(L,K)=(4.0D0**(L-1)*T(L-1,K+1)-T(L-1,K))/
$      (4.0D0**(L-1)-1.0D0)
200  CONTINUE
C
C-----CHECK ACCURACY CRITERION
C
      IF(T(J,1).EQ. 0.0D0) THEN
          RESULT=T(J,1)
          GO TO 111
      END IF
C
      IF(DABS((T(J,1)-T(J-1,1))/T(J,1)).GE. EPS) THEN
          J=J+1
          IF(J.GT.MAX) THEN
              PAUSE 'TOO MANY STEPS'
              GO TO 111
          
```

```

        ELSE
            GO TO 50
        END IF
    ELSE
        RESULT=T(J,1)
    END IF
111 RETURN
222 END
C
C*****C
C    SUBPROGRAM                                C
C    ERROR FUNCTION                            C
C*****C
    FUNCTION ERF(X)
    IMPLICIT DOUBLE PRECISION(A-H,0-Z)
    IF(X.LT.0.0D0)THEN
        ERF=-GAMMP(.5D0,X**2)
    ELSE
        ERF=GAMMP(.5D0,X**2)
    ENDIF
    RETURN
    END
C
C*****C
    FUNCTION ERFC(X)
    IMPLICIT DOUBLE PRECISION(A-H,0-Z)
    IF(X.LT.0.0D0)THEN
        ERFC=1.0D0+GAMMP(.5D0,X**2)
    ELSE
        ERFC=GAMMQ(.5D0,X**2)
    ENDIF
    RETURN
    END
C*****C
    FUNCTION GAMMP(A,X)
    IMPLICIT DOUBLE PRECISION(A-H,0-Z)
    IF(X.LT.0.0D0 .OR. A.LE.0.0D0)PAUSE
    IF(X.LT.A+1.0D0)THEN
        CALL GSER(GAMSER,A,X,GLN)
        GAMMP=GAMSER
    ELSE
        CALL GCF(GAMMCF,A,X,GLN)
        GAMMP=1.0D0-GAMMCF
    ENDIF
    RETURN
    END
C*****C
    FUNCTION GAMMQ(A,X)
    IMPLICIT DOUBLE PRECISION(A-H,0-Z)
    IF(X.LT.0.0D0 .OR. A.LE.0.0D0)PAUSE
    IF(X.LT.A+1.0D0)THEN
        CALL GSER(GAMSER,A,X,GLN)
        GAMMQ=1.0D0-GAMSER
    ELSE

```

```

        CALL GCF(GAMMCF,A,X,GLN)
        GAMMQ=GAMMCF
    ENDIF
    RETURN
END
C*****C
    SUBROUTINE GCF(GAMMCF,A,X,GLN)
    IMPLICIT DOUBLE PRECISION(A-H,O-Z)
    PARAMETER (ITMAX=200,EPS=3.D-12)
    GLN=GAMMLN(A)
    GOLD=0.0D0
    AO=1.0D0
    A1=X
    BO=0.0D0
    B1=1.0D0
    FAC=1.0D0
    DO 11 N=1,ITMAX
        AN=DFLOAT(N)
        ANA=AN-A
        AO=(A1+AO*ANA)*FAC
        BO=(B1+BO*ANA)*FAC
        ANF=AN*FAC
        A1=X*AO+ANF*A1
        B1=X*BO+ANF*B1
        IF(A1.NE.0.0D0)THEN
            FAC=1.0D0/A1
            G=B1*FAC
            IF(DABS((G-GOLD)/G).LT.EPS)GO TO 1
            GOLD=G
        ENDIF
11    CONTINUE
    PAUSE 'A TOO LARGE, ITMAX TOO SMALL'
1    GAMMCF=DEXP(-X+A*DLOG(X)-GLN)*G
    RETURN
    END
C*****C
    SUBROUTINE GSER(GAMSER,A,X,GLN)
    IMPLICIT DOUBLE PRECISION(A-H,O-Z)
    PARAMETER (ITMAX=200,EPS=3.D-12)
    GLN=GAMMLN(A)
    IF(X.LE.0.0D0)THEN
        IF(X.LT.0.0D0)PAUSE 'X LT 0.0D0'
        GAMSER=0.0D0
        RETURN
    ENDIF
    AP=A
    SUM=1.0D0/A
    DEL=SUM
    DO 11 N=1,ITMAX
        AP=AP+1.0D0
        DEL=DEL*X/AP
        SUM=SUM+DEL
        IF(DABS(DEL).LT.DABS(SUM)*EPS)GO TO 1
11    CONTINUE

```

```

        PAUSE 'A TOO LARGE, ITMAX TOO SMALL'
1      GAMSER=SUM*DEXP(-X+A*DLOG(X)-GLN)
        RETURN
        END
C*****C
        FUNCTION GAMMLN(XX)
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        DIMENSION COF(6)
        DATA COF,STP/76.180091729406D0,-86.505320327112D0,
$      24.014098222230D0,-1.231739516140D0,.120858003D-2,
$      -.536382D-5,2.50662827465D0/
        DATA HALF,ONE,FPF/0.5D0,1.0D0,5.5D0/
        X=XX-ONE
        TMP=X+FPF
        TMP=(X+HALF)*DLOG(TMP)-TMP
        SER=ONE
        DO 11 J=1,6
            X=X+ONE
            SER=SER+COF(J)/X
11      CONTINUE
        GAMMLN=TMP+DLOG(STP*SER)
        RETURN
        END
C*****C
C      USING BRENT'S METHOD                                C
C      FOR FINDING THE ROOT OF FUNCTION TFUNC              C
C*****C
        FUNCTION ZBRENT(TFUNC,X1,X2,TOL)
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        EXTERNAL TFUNC1
        PARAMETER (ITMAX=100,EPS=3.D-16)
        A=X1
        B=X2
        FA=TFUNC(A)
        FB=TFUNC(B)
        IF(FB*FA.GT.0.D0) PAUSE 'ROOT MUST BE BRACKETED FOR ZBRENT.'
        FC=FB
        DO 11 ITER=1,ITMAX
            IF(FB*FC.GT.0.D0) THEN
                C=A
                FC=FA
                D=B-A
                E=D
            ENDIF
            IF(DABS(FC).LT.DABS(FB)) THEN
                A=B
                B=C
                C=A
                FA=FB
                FB=FC
                FC=FA
            ENDIF
            TOL1=2.D0*EPS*DABS(B)+0.5D0*TOL
            XM=.5D0*(C-B)

```



```

      IF(DABS(XM).LE.TOL1 .OR. FB.EQ.0.DO)THEN
        ZBRENT=B
        RETURN
      ENDIF
      IF(DABS(E).GE.TOL1 .AND. DABS(FA).GT.DABS(FB)) THEN
        S=FB/FA
        IF(A.EQ.C) THEN
          P=2.DO*XM*S
          Q=1.DO-S
        ELSE
          Q=FA/FC
          R=FB/FC
          P=S*(2.DO*XM*Q*(Q-R)-(B-A)*(R-1.DO))
          Q=(Q-1.DO)*(R-1.DO)*(S-1.DO)
        ENDIF
        IF(P.GT.0.DO) Q=-Q
        P=DABS(P)
        IF(2.DO*P .LT. DMIN1(3.*XM*Q-DABS(TOL1*Q),DABS(E*Q)))
$      THEN
          E=D
          D=P/Q
        ELSE
          D=XM
          E=D
        ENDIF
      ELSE
        D=XM
        E=D
      ENDIF
      A=B
      FA=FB
      IF(DABS(D) .GT. TOL1) THEN
        B=B+D
      ELSE
        B=B+DSIGN(TOL1,XM)
      ENDIF
      FB=TFUNC(B)
11  CONTINUE
      PAUSE 'ZBRENT EXCEEDING MAXIMUM ITERATIONS.'
      ZBRENT=B
      RETURN
      END
C
C*****C
      SUBROUTINE ZBRAK(FX,X1,X2,N,XB1,XB2,NB)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION XB1(1),XB2(1)
      NBB=NB
      NB=0
      X=X1
      DX=(X2-X1)/N
      FP=FX(X)
      DO 11 I=1,N
        X=X+DX

```



```
FC=FX(X)
IF(FC*FP.LT.0.) THEN
  NB=NB+1
  XB1(NB)=X-DX
  XB2(NB)=X
ENDIF
FP=FC
IF(NBB.EQ.NB)RETURN
11 CONTINUE
RETURN
END
```

APPENDIX G
SSM FORTRAN PROGRAM FOR COMBINATION PROBLEM

```

C*****C
C      SOLUTION OF 1D ABLATION PROBLEM WITH TWO MOVING BOUNDARIES      C
C      IN A SEMI-INFINITE MEDIUM WITH SUBCOOLING BY THE SOURCE-        C
C      AND-SINK METHOD                                                  C
C                                                                           C
C      INPUT FILE UNIT NUMBER IS SET AT 15                             C
C      OUTPUT FILE UNIT IS SET AT 16                                    C
C                                                                           C
C              NOTATIONS                                                C
C                                                                           C
C      TMI      =TIME WHEN PHASE CHANGE STARTS                          C
C      DELTM    =TIME-STEP SIZE                                         C
C      ALPHA    =THERMAL DIFFUSIVITY                                    C
C      CK       =THERMAL CONDUCTIVITY                                    C
C      CP       =SPECIFIC HEAT                                          C
C      CLF      =LATENT HEAT OF FUSION                                   C
C      CLV      =LATENT HEAT OF VAPORIZATION                            C
C      TIN      =TEMPERATURE AT INTERIOR POINTS                         C
C      R1       =SOLID-LIQUID INTERFACE POSITION                          C
C      R2       =ABLATED SURFACE POSITION                                C
C      R1V      =SOLID-LIQUID INTERFACE VELOCITY                       C
C      R2V      =SURFACE VELOCITY                                       C
C      RHO      =DENSITY                                                 C
C      TM       =MELTING TEMPERATURE                                    C
C      TM       =VAPORIZATION TEMPERATURE                              C
C*****C
C
C-----MAIN PROGRAM
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER(MAXF=2500,NP=3,PI=3.141592653589793D0)
      COMMON /T/NF,INF,TMI,TMF,DELT,TRTM
      PARAMETER (TOLX=1.0D-12)
      PARAMETER (N=3,TOLF=1.0D-12)
      COMMON /C/ALPHA
      DIMENSION XX(NP)
      COMMON /C1/CK,RHO,CP,CLF,CLV,TM,TV
      COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
      COMMON /R2/R2(MAXF),RV2(MAXF),MF,MT
      COMMON /RINT/RIX
      COMMON /DIF/CKA,CKB,DIFF
      COMMON /HEAT1/RX(300),TIN(300)
      COMMON/HEAT/ERRX,ERRF
      EXTERNAL TFUNC1,TFUNC2,FUNC12,FUNC22A,TFUNC3,FUNC1
      EXTERNAL FUNC3B,FUNC3A,FUNC2A,FUNC2B,FUNC11,TFUNCA

```

C

```

READ(15,*) CK,RHO,CP,CLF,CLV,TM,TV
WRITE(*,*)'TMI,EPSILON,DELTM='
READ(*,*) TMI,EPSILON,DELTM
WRITE(*,*)'TOTAL NO. OF ITERATION TIME STEP MF=?'
READ(*,*)MF
MT=10000
WRITE(*,*)'NO. OF INTERNAL COMPUTATION STEP NPS=?'
READ(*,*)NPS
WRITE(16,*)'-----'
WRITE(16,*)'INPUT DATA'
WRITE(16,111)TM
WRITE(16,122)TV
WRITE(16,133)CK
WRITE(16,144)RHO
WRITE(16,155)CP
WRITE(16,166)CLF
WRITE(16,177)CLV
WRITE(16,188)DELTM
WRITE(16,*)'-----'
WRITE(16,*)'      TIME ', '      INTERFACE'
WRITE(16,*)' '
WRITE(16,*)'      ', '      R1 ', '      R2 ', '
$      G'
WRITE(77,*)'      TIME      X      TEMP'

```

C

C-----INITIALIZE R1,R2,RV1,RV2,RX,TIN

C

```

WRITE(66,*)' '
WRITE(66,*)'      CKA=RHO*LV*DR2/DT'
WRITE(66,*)'      CKB=-K*DT(R2,T)/DT'
WRITE(66,*)'-----'
WRITE(66,*)' '
WRITE(66,*)'      TIME', '      G(T) ', '      CKA', '
$      CKB', '      G(T)-(CKA+CKB)'
WRITE(66,*)' '
DO I=1,MAXF
  R1(I)=0.0D0
  RV1(I)=0.0D0
  R2(I)=0.0D0
  RV2(I)=0.0D0
  Q(NF)=0.0D0
ENDDO
DO I=1,300
  RX(I)=0.0D0
  TIN(I)=0.0D0
ENDDO

```

C

ALPHA=CK/RHO/CP

C

```

DO 100 NF=1,MF
  XNF=NF
  TFF=TMI+XNF*DELTM

```

C

```

      TOL=1.D-15
      IF(NF .LT. MT)THEN
C
C-----ASSUME INTERFACE POSITION INTERVAL R1(NF)
C
      IF(NF .EQ. 1) THEN
        WRITE(*,*)TFF,' ENTER INITIAL RA,RB=?'
        READ(*,*) RA,RB
      ELSE
        RA=R1(NF-1)/2.0D0
        RB=5.0D0*R1(NF-1)
      END IF
C
88      R1(NF)=ZBRENT(TFUNCA,RA,RB,TOL)
      AA=DSQRT(TFF)
      CALL ROMBERG(FUNC12,AA,0.0D0,QUICK)
      QUICK=QUICK*DSQRT(ALPHA/PI)/CK
      SUMP=0.0D0
      DO INF=1,NF
        TF2=TMI+INF*DELTM
        TF1=TMI+(INF-1)*DELTM
        AA=DSQRT(TFF-TF1)
        BB=DSQRT(TFF-TF2)
        CALL ROMBERG(FUNC22A,AA,BB,P)
        P=P*RV1(INF)
        SUMP=SUMP+P
      ENDDO
      SUMP=SUMP*CLF/2.0D0/CP/DSQRT(PI*ALPHA)
      SP=+SUMP-QUICK+TV
      WRITE(*,*)SP
      IF(SP .LE. EPSILON)THEN
        MT=NF
        WRITE(*,*) NF,R1(NF)
        WRITE(16,1000)TFF,R1(NF)
C-----UPDATE INTERFACE POSITION VELOCITY
C
      IF(NF .EQ. 1) THEN
        RV1(NF)=R1(NF)/DELTM
      ELSE
        RV1(NF)=(R1(NF)-R1(NF-1))/DELTM
      END IF
C
C-----INTERNAL POINT TEMP. COMPUTATION
C
      IF(MOD(NF,NPS) .NE. 0) GO TO 100
      RIX=0.0D0
      RX(1)=RIX
      CALL INTERNAL(TINTL)
      TIN(1)=TINTL
      DO I=1,10
        RIX=R1(NF)/10.0D0*I
        RX(I+1)=RIX
        CALL INTERNAL(TINTL)
        TIN(I+1)=TINTL

```

```

      ENDDO
      DO I=1,42
        RIX=R1(NF)+R1(NF)/2.0D0*I
        RX(I+11)=RIX
        CALL INTERNAL(TINTL)
        TIN(I+11)=TINTL
      ENDDO

C
C-----COMPUTATION OF STORED HEAT
C
      CALL HEATS
      GO TO 100
    ELSE
C-----UPDATE INTERFACE POSITION VELOCITY
C
      IF(NF .EQ. 1) THEN
        RV1(NF)=R1(NF)/DELTM
      ELSE
        RV1(NF)=(R1(NF)-R1(NF-1))/DELTM
      END IF

C
C-----INTERNAL POINT TEMP. COMPUTATION
C
      IF(MOD(NF,NPS) .NE. 0) GO TO 100
      RIX=0.0D0
      RX(1)=RIX
      CALL INTERNAL(TINTL)
      TIN(1)=TINTL
      DO I=1,10
        RIX=R1(NF)/10.0D0*I
        RX(I+1)=RIX
        CALL INTERNAL(TINTL)
        TIN(I+1)=TINTL
      ENDDO
      DO I=1,42
        RIX=R1(NF)+R1(NF)/2.0D0*I
        RX(I+11)=RIX
        CALL INTERNAL(TINTL)
        TIN(I+11)=TINTL
      ENDDO

C
C-----COMPUTATION OF STORED HEAT
C
      CALL HEATS
      GO TO 100
    ENDIF
  ELSE
    TRTM=TMI+MT*DELTM
    IF(NF .EQ. MT+1) THEN

C
C-----COMPUTE TWO INTERFACE POSITIONS R1,R2
C-----INITIAL GUESS OF INTERFACE POSITION R2(NF)
C
      WRITE(*,*)NF,' ENTER ASSUMED INITIAL R1=? , R2=? AND Q'

```

```

      READ(*,*)R1(NF),R2(NF),Q(NF)
    ELSE
      R2(NF)=1.002D0*R2(NF-1)
      R1(NF)=1.002D0*R1(NF-1)
      Q(NF)=1.001D0*Q(NF-1)
    ENDIF
C-----CREATE VECTOR XX
C
      XX(1)=R1(NF)
      XX(2)=R2(NF)
      XX(3)=Q(NF)
C
C-----COMPUTE B(NF) AND Q(NF)
C
      NTRIAL=1000
      CALL MNEWT(NTRIAL,XX,N,TOLX,TOLF)
C
      WRITE(16,1200)TFF,R1(NF),R2(NF),Q(NF)
C
C-----UPDATE INTERFACE POSITION VELOCITY
C
      RV1(NF)=(R1(NF)-R1(NF-1))/DELTM
      IF(NF.EQ. MT+1) THEN
        RV2(NF)=R2(NF)/DELTM
      ELSE
        RV2(NF)=(R2(NF)-R2(NF-1))/DELTM
      END IF
C
C-----INTERNAL POINTS TEMP. COMPUTATION
C
      IF(MOD(NF,NPS).NE. 0) GO TO 100
      RIX=0.0D0
      RX(1)=RIX
      CALL INTERNAL1(TINTL)
      TIN(1)=TINTL
      DO I=1,10
        RIX=R2(NF)/10.0D0*I
        RX(I+1)=RIX
        CALL INTERNAL1(TINTL)
        TIN(I+1)=TINTL
      ENDDO
      DO I=1,50
        RIX=(R1(NF)-R2(NF))/50.0D0*I+R2(NF)
        RX(I+11)=RIX
        CALL INTERNAL1(TINTL)
        TIN(I+11)=TINTL
      ENDDO
      DO I=1,30
        RIX=R1(NF)+R1(NF)/4.0D0*I
        RX(I+61)=RIX
        CALL INTERNAL1(TINTL)
        TIN(I+61)=TINTL
      ENDDO
      CALL HEATS

```

```

        ENDIF
100  CONTINUE
    WRITE(16,199)TRTM
C
C-----FORMATS
C
111  FORMAT(// 'MELTING TEMPERATURE=',F7.2)
122  FORMAT(' VAPORIZATION TEMPERATURE=',F7.2)
133  FORMAT(' THERMAL CONDUCTIVITY=',F10.6)
144  FORMAT(' DENSITY=',F10.6)
155  FORMAT(' SPECIFIC HEAT CAPACITY=',F10.6)
166  FORMAT(' FUSION LATENT HEAT=',F10.7)
177  FORMAT(' VAPORIZATION LATENT HEAT=',F10.7)
188  FORMAT('/ TIME STEP SIZE=',F10.7)
199  FORMAT(' TRANSIENT TIME=',F10.7//)
1000 FORMAT(3X,F13.7,5X,E14.7)
1200 FORMAT(2X,F13.7,5X,E14.7,5X,E14.7,5X,E14.7)
1300 FORMAT(2X,F10.7,2X,F14.7,2X,F14.7,2X,F14.7,5X,E14.7)
    END
C
C*****C
C    SUBPROGRAM MNEWTON                                C
C*****C
    SUBROUTINE MNEWT(NTRIAL,XX,N,TOLX,TOLF)
    IMPLICIT DOUBLE PRECISION(A-H,O-Z)
    PARAMETER (NP=3)
    PARAMETER (MAXF=2500,AD=1.1D0,J=2)
    COMMON/HEAT/ERRX,ERRF
    DIMENSION ALPHA1(NP,NP),BETA(NP),INDX(NP),XX(NP)
C
    DO 13 K=1,NTRIAL
        CALL USRFUN(XX,ALPHA1,BETA)
        ERRF=0.0D0
        DO 11 I=1,N
            ERRF=ERRF+DABS(BETA(I))
11        CONTINUE
            IF(ERRF.LE.TOLF)RETURN
            CALL LUDCMP(ALPHA1,N,NP,INDX,D)
            CALL LUBKSB(ALPHA1,N,NP,INDX,BETA)
            ERRX=0.0D0
            DO 12 I=1,N
                ERRX=ERRX+DABS(BETA(I))
                XX(I)=XX(I)+BETA(I)
12        CONTINUE
            IF(ERRX.LE.TOLX)RETURN
13    CONTINUE
        RETURN
    END
C*****C
C    SUBPROGRAM USERFUNCTION                                C
C*****C
    SUBROUTINE USRFUN(XX,ALPHA1,BETA)
    IMPLICIT DOUBLE PRECISION(A-H,O-Z)
    PARAMETER (MAXF=2500,PI=3.141592653589793D0,NP=3)

```



```

DIMENSION ALPHA1(NP,NP),BETA(NP),XX(NP)
COMMON /T/NF,INF,TMI,TMF,DELTMT,TRTM
COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
COMMON /R2/R2(MAXF),RV2(MAXF),MF,MT

C
CH=1.D-10

C
R1(NF)=XX(1)
R2(NF)=XX(2)
Q(NF)=XX(3)

C
F1=TFUNC1(R1(NF))
R1(NF)=R1(NF)+CH
ALPHA1(1,1)=(TFUNC1(R1(NF))-F1)/CH
R1(NF)=R1(NF)-CH
R2(NF)=R2(NF)+CH
ALPHA1(1,2)=(TFUNC1(R1(NF))-F1)/CH
R2(NF)=R2(NF)-CH
Q(NF)=Q(NF)+CH
ALPHA1(1,3)=(TFUNC1(R1(NF))-F1)/CH
Q(NF)=Q(NF)-CH

C
F2=TFUNC2(R2(NF))
R2(NF)=R2(NF)+CH
ALPHA1(2,2)=(TFUNC2(R2(NF))-F2)/CH
R2(NF)=R2(NF)-CH
R1(NF)=R1(NF)+CH
ALPHA1(2,1)=(TFUNC2(R2(NF))-F2)/CH
R1(NF)=R1(NF)-CH
Q(NF)=Q(NF)+CH
ALPHA1(2,3)=(TFUNC2(R2(NF))-F2)/CH
Q(NF)=Q(NF)-CH

C
F3=TFUNC3(Q(NF))
Q(NF)=Q(NF)+CH
ALPHA1(3,3)=(TFUNC3(Q(NF))-F3)/CH
Q(NF)=Q(NF)-CH
R1(NF)=R1(NF)+CH
ALPHA1(3,1)=(TFUNC3(Q(NF))-F3)/CH
R1(NF)=R1(NF)-CH
R2(NF)=R2(NF)+CH
ALPHA1(3,2)=(TFUNC3(Q(NF))-F3)/CH
R2(NF)=R2(NF)-CH

C
BETA(1)=-F1
BETA(2)=-F2
BETA(3)=-F3

C
END

C*****C
C SUBPROGRAM TFUNCA C
C*****C
FUNCTION TFUNCA(X)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)

```

```

PARAMETER(MAXF=2500,PI=3.141592653589793D0)
COMMON /T/NF,INF,TMI,TMF,DELTM,TRTM
COMMON /C/ALPHA
COMMON /C1/CK,RHO,CP,CLF,CLV,TM,TV
COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
COMMON /R2/R2(MAXF),RV2(MAXF),MF,MT
COMMON/REST/QINTT,SUMA,SUMB,SUMC,SUMAA
EXTERNAL FUNC1,FUNC2A,FUNC2B
R1(NF)=X

```

C

```

TFF2=TMI+NF*DELTM
TFF1=TMI+(NF-1)*DELTM
IF(NF .EQ. 1) THEN
    RV1(NF)=R1(NF)/(TFF2-TFF1)
ELSE
    RV1(NF)=(R1(NF)-R1(NF-1))/(TFF2-TFF1)
END IF

```

C

```

IF(TMI .EQ. 0.0D0)THEN
    QINT=0.0D0
ELSE
    AA=DSQRT(TFF2)
    BB=DSQRT(TFF2-TMI)
    CALL ROMBERG(FUNC1,AA,BB,QINT)
ENDIF

```

C

```

QINT1=0.0D0
DO INF=1,NF
    TF2=TMI+INF*DELTM
    TF1=TMI+(INF-1)*DELTM
    AA=DSQRT(TFF2-TF1)
    BB=DSQRT(TFF2-TF2)
    CALL ROMBERG(FUNC1,AA,BB,A)
    QINT1=QINT1+A
ENDDO

```

C

```

SUMB1=0.0D0
SUMB2=0.0D0
DO INF=1,NF
    TF2=TMI+INF*DELTM
    TF1=TMI+(INF-1)*DELTM
    AA=DSQRT(TFF2-TF1)
    BB=DSQRT(TFF2-TF2)
    CALL ROMBERG(FUNC2A,AA,BB,B1)
    CALL ROMBERG(FUNC2B,AA,BB,B2)
    B1=RV1(INF)*B1
    B2=RV1(INF)*B2
    SUMB1=SUMB1+B1
    SUMB2=SUMB2+B2
ENDDO
SUMB=SUMB1+SUMB2

```

C

```

TFUNCA=(2.0D0*ALPHA/CK*(QINT+QINT1)-CLF/CP*SUMB)
$      /2.0D0/DSQRT(PI*ALPHA)-TM

```

```

RETURN
END
C
C*****C
C    SUBPROGRAM TFUNC1                                C
C*****C
      FUNCTION TFUNC1(X)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER(MAXF=2500,PI=3.141592653589793D0)
      COMMON /T/NF,INF,TMI,TMF,DELTM,TRTM
      COMMON /C/ALPHA
      COMMON /C1/CK,RH0,CP,CLF,CLV,TM,TV
      COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
      COMMON /R2/R2(MAXF),RV2(MAXF),MF,MT
      EXTERNAL FUNC1,FUNC4,FUNC5A,FUNC5B,FUNC6A,FUNC6B
      $      ,FUNC2A,FUNC2B,FUNC3A,FUNC3B,FUNC44,FUNC11
      R1(NF)=X
C
      TFF2=TMI+NF*DELTM
      TFF1=TMI+(NF-1)*DELTM
      IF(NF .EQ. MT+1) THEN
        RV2(NF)=R2(NF)/(TFF2-TFF1)
      ELSE
        RV2(NF)=(R2(NF)-R2(NF-1))/(TFF2-TFF1)
      END IF
C
      RV1(NF)=(R1(NF)-R1(NF-1))/(TFF2-TFF1)
C
      AA=DSQRT(TFF2)
      BB=DSQRT(TFF2-TRTM)
      CALL ROMBERG(FUNC1,AA,BB,QINTT)
C
      SUMA=0.0D0
      DO INF=MT+1,NF
        TF2=TMI+INF*DELTM
        TF1=TMI+(INF-1)*DELTM
        AA=DSQRT(TFF2-TF1)
        BB=DSQRT(TFF2-TF2)
        CALL ROMBERG(FUNC11,AA,BB,A)
        A=A*Q(INF)
        SUMA=SUMA+A
      ENDDO
C
      SUMB1=0.0D0
      SUMB2=0.0D0
      DO INF=1,NF
        TF2=TMI+INF*DELTM
        TF1=TMI+(INF-1)*DELTM
        AA=DSQRT(TFF2-TF1)
        BB=DSQRT(TFF2-TF2)
        CALL ROMBERG(FUNC2A,AA,BB,B1)
        CALL ROMBERG(FUNC2B,AA,BB,B2)
        B1=RV1(INF)*B1
        B2=RV1(INF)*B2

```

```

        SUMB1=SUMB1+B1
        SUMB2=SUMB2+B2
ENDDO
SUMB=SUMB1+SUMB2
C
SUMC1=0.0D0
SUMC2=0.0D0
DO INF=MT+1,NF
    TFF2=TMI+INF*DELTM
    TFF1=TMI+(INF-1)*DELTM
    AA=DSQRT(TFF2-TFF1)
    BB=DSQRT(TFF2-TFF2)
    CALL ROMBERG(FUNC3A,AA,BB,C1)
    CALL ROMBERG(FUNC3B,AA,BB,C2)
    C1=RV2(INF)*C1
    C2=RV2(INF)*C2
    SUMC1=SUMC1+C1
    SUMC2=SUMC2+C2
ENDDO
SUMC=SUMC1+SUMC2
C
TFUNC1=TM-DSQRT(ALPHA/PI)/CK*(QINTT+SUMA)+1.0D0/2.0D0/CP/
$    DSQRT(PI*ALPHA)*(CLF*SUMB+CLV*SUMC)
RETURN
END
C*****C
C    SUBPROGRAM TFUNCT2                                C
C*****C
FUNCTION TFUNCT2(X)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
PARAMETER(MAXF=2500,PI=3.141592653589793D0)
COMMON /T/NF,INF,TMI,TMF,DELTM,TRTM
COMMON /C/ALPHA
COMMON /C1/CK,RHO,CP,CLF,CLV,TM,TV
COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
COMMON /R2/R2(MAXF),RV2(MAXF),MF,MT
EXTERNAL FUNC4,FUNC44,FUNC5A,FUNC5B,FUNC6A,FUNC6B
R2(NF)=X
C
TFF2=TMI+NF*DELTM
TFF1=TMI+(NF-1)*DELTM
IF(NF.EQ.MT+1) THEN
    RV2(NF)=R2(NF)/(TFF2-TFF1)
ELSE
    RV2(NF)=(R2(NF)-R2(NF-1))/(TFF2-TFF1)
END IF
C
RV1(NF)=(R1(NF)-R1(NF-1))/(TFF2-TFF1)
C
AA=DSQRT(TFF2)
BB=DSQRT(TFF2-TRTM)
CALL ROMBERG(FUNC4,AA,BB,QINTTT)
C
SUMD=0.0D0

```

```

DO INF=MT+1,NF
  TF2=TMI+INF*DELTM
  TF1=TMI+(INF-1)*DELTM
  AA=DSQRT(TFF2-TF1)
  BB=DSQRT(TFF2-TF2)
  CALL ROMBERG(FUNC44,AA,BB,D)
  D=D*Q(INF)
  SUMD=SUMD+D
ENDDO

```

C

```

SUME1=0.0D0
SUME2=0.0D0
DO INF=1,NF
  TF2=TMI+INF*DELTM
  TF1=TMI+(INF-1)*DELTM
  AA=DSQRT(TFF2-TF1)
  BB=DSQRT(TFF2-TF2)
  CALL ROMBERG(FUNC5A,AA,BB,E1)
  CALL ROMBERG(FUNC5B,AA,BB,E2)
  E1=RV1(INF)*E1
  E2=RV1(INF)*E2
  SUME1=SUME1+E1
  SUME2=SUME2+E2
ENDDO
SUME=SUME1+SUME2

```

C

```

SUMF1=0.0D0
SUMF2=0.0D0
DO INF=MT+1,NF
  TF2=TMI+INF*DELTM
  TF1=TMI+(INF-1)*DELTM
  AA=DSQRT(TFF2-TF1)
  BB=DSQRT(TFF2-TF2)
  CALL ROMBERG(FUNC6A,AA,BB,F1)
  CALL ROMBERG(FUNC6B,AA,BB,F2)
  F1=RV2(INF)*F1
  F2=RV2(INF)*F2
  SUMF1=SUMF1+F1
  SUMF2=SUMF2+F2
ENDDO
SUMF=SUMF1+SUMF2

```

C

```

TFUNC2=TV-DSQRT(ALPHA/PI)/CK*(QINTTT+SUMD)+1.0D0/2.0D0/CP/
$ DSQRT(PI*ALPHA)*(CLF*SUME+CLV*SUMF)
RETURN
END

```

C*****C

C SUBPROGRAM TFUNC3 C

C*****C

```

FUNCTION TFUNC3(X)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
PARAMETER(MAXF=2500,PI=3.141592653589793D0)
COMMON /T/NF,INF,TMI,TMF,DELTM,TRTM
COMMON /C/ALPHA

```

```

COMMON /DIF/CKA,CKB,DIFF
COMMON /C1/CK,RH0,CP,CLF,CLV,TM,TV
COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
COMMON /R2/R2(MAXF),RV2(MAXF),MF,MT
EXTERNAL FUNC7,FUNC8
Q(NF)=X

```

C

```

TFF2=TMI+NF*DELTM
TFF1=TMI+(NF-1)*DELTM
IF(NF .EQ. MT+1) THEN
    RV2(NF)=R2(NF)/(TFF2-TFF1)
ELSE
    RV2(NF)=(R2(NF)-R2(NF-1))/(TFF2-TFF1)
END IF

```

C

```
RV1(NF)=(R1(NF)-R1(NF-1))/(TFF2-TFF1)
```

C

```
CALL ROMBERG(FUNC7,0.0D0,TRTM,QINTIV)
```

C

```

SUMG=0.0D0
DO INF=MT+1,NF
    TF2=TMI+INF*DELTM
    TF1=TMI+(INF-1)*DELTM
    IF(INF .EQ. NF) THEN
        A1=2.0D0*DSQRT(ALPHA*PI)/R2(NF)
        A2=R2(NF)/DSQRT(4.0D0*ALPHA*(TFF2-TFF1))
        G=A1*ERFC(A2)
    ELSE
        A1=4.0D0*DSQRT(ALPHA)/R2(NF)
        A2=R2(NF)/DSQRT(4.0D0*ALPHA*(TFF2-TF1))
        A3=R2(NF)/DSQRT(4.0D0*ALPHA*(TFF2-TF2))
        CALL ROMBERG(FUNC8,A2,A3,G)
        G=A1*G
    ENDIF
    G=Q(INF)*G
    SUMG=SUMG+G
ENDDO

```

C

```

SUMH1=0.0D0
SUMH2=0.0D0
DO INF=1,NF
    TF2=TMI+INF*DELTM
    TF1=TMI+(INF-1)*DELTM
    IF(INF .EQ. 1) THEN
        AM=RV1(INF)
        BN=-R1(INF)*TF1/(TF2-TF1)
    ELSE
        AM=RV1(INF)
        BN=(R1(INF-1)*TF2-R1(INF)*TF1)/(TF2-TF1)
    ENDIF
    IF(INF .EQ. NF) THEN
        CC=(R2(NF)-2.0D0*AM*TFF2+AM*TF1-BN)/
$          DSQRT(4.0D0*ALPHA*(TFF2-TF1))
        C01=DEXP(-AM*(R2(NF)-AM*TFF2-BN)/ALPHA)/2.0D0
        H1=C01*(-1.0D0-ERF(CC))
    
```



```

C
$      FF=(R2(NF)+2.0D0*AM*TFF2-AM*TF1+BN)/
      DSQRT(4.0D0*ALPHA*(TFF2-TF1))
C02=DEXP(AM*(R2(NF)+AM*TFF2+BN)/ALPHA)/2.0D0
IF((R2(NF)+(AM*TFF2+BN)) .LT.0.0D0)THEN
      H2=C02*(-1.0D0-ERF(FF))
ELSE
      H2=C02*(1.0D0-ERF(FF))
ENDIF
ELSE
C01=DEXP(-AM*(R2(NF)-AM*TFF2-BN)/ALPHA)/DSQRT(PI)
CC=(R2(NF)-2.0D0*AM*TFF2+AM*TF1-BN)/
$      DSQRT(4.0D0*ALPHA*(TFF2-TF1))
CC1=(R2(NF)-2.0D0*AM*TFF2+AM*TF2-BN)/
$      DSQRT(4.0D0*ALPHA*(TFF2-TF2))
CALL ROMBERG(FUNC8,CC,CC1,H1)
H1=C01*H1
C
C02=DEXP(AM*(R2(NF)+AM*TFF2+BN)/ALPHA)/DSQRT(PI)
FF=(R2(NF)+2.0D0*AM*TFF2-AM*TF1+BN)/
$      DSQRT(4.0D0*ALPHA*(TFF2-TF1))
FF1=(R2(NF)+2.0D0*AM*TFF2-AM*TF2+BN)/
$      DSQRT(4.0D0*ALPHA*(TFF2-TF2))
CALL ROMBERG(FUNC8,FF,FF1,H2)
H2=C02*H2
ENDIF
H1=RV1(INF)*H1
H2=RV1(INF)*H2
SUMH1=SUMH1+H1
SUMH2=SUMH2+H2
ENDDO
SUMH=SUMH1+SUMH2
C
SUMI1=0.0D0
SUMI2=0.0D0
DO INF=MT+1,NF
  TF2=TMI+INF*DELTM
  TF1=TMI+(INF-1)*DELTM
  IF(NF .EQ. MT+1) THEN
    AM=RV2(INF)
    BN=-R2(INF)*TF1/(TF2-TF1)
  ELSE
    AM=RV2(INF)
    BN=(R2(INF-1)*TF2-R2(INF)*TF1)/(TF2-TF1)
  ENDIF
  IF(INF .EQ. NF)THEN
    CI1=-ERF(-AM*(TFF2-TFF1)/DSQRT
$      (4.0D0*ALPHA*(TFF2-TFF1)))/2.0D0
    FF=(R2(NF)+2.0D0*AM*TFF2-AM*TF1+BN)/
$      DSQRT(4.0D0*ALPHA*(TFF2-TF1))
    C02=DEXP(AM*(R2(NF)+R2(NF))/ALPHA)/2.0D0
    CI2=C02*(1.0D0-ERF(FF))
  ELSE
    C01=DEXP(-AM*(R2(NF)-AM*TFF2-BN)/ALPHA)/DSQRT(PI)

```



```

      CC=(R2(NF)-2.0D0*AM*TFF2+AM*TF1-BN)/
$      DSQRT(4.0D0*ALPHA*(TFF2-TF1))
      CC1=(R2(NF)-2.0D0*AM*TFF2+AM*TF2-BN)/
$      DSQRT(4.0D0*ALPHA*(TFF2-TF2))
      CALL ROMBERG(FUNC8,CC,CC1,CI1)
      CI1=C01*CI1
      C02=DEXP(AM*(R2(NF)+AM*TFF2+BN)/ALPHA)/DSQRT(PI)
      FF=(R2(NF)+2.0D0*AM*TFF2-AM*TF1+BN)/
$      DSQRT(4.0D0*ALPHA*(TFF2-TF1))
      FF1=(R2(NF)+2.0D0*AM*TFF2-AM*TF2+BN)/
$      DSQRT(4.0D0*ALPHA*(TFF2-TF2))
      CALL ROMBERG(FUNC8,FF,FF1,CI2)
      CI2=C02*CI2
      ENDIF
      CI1=RV2(INF)*CI1
      CI2=RV2(INF)*CI2
      SUMI1=SUMI1+CI1
      SUMI2=SUMI2+CI2
      ENDDO
      SUMI=SUMI1+SUMI2
      TFUNC3=GX(TFF2)-R2(NF)/2.0D0/DSQRT(PI*ALPHA)*
$      (QINTIV+SUMG)+RHO*(CLF*SUMH+CLV*SUMI-CLV*RV2(NF))
      RETURN
      END

```

C*****C

```

      FUNCTION FUNC1(X)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER(MAXF=2500)
      COMMON /T/NF,INF,TMI,TMF,DELT,TRTM
      COMMON /C/ALPHA
      COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
      TFF=TMI+NF*DELT
      IF(X.EQ. 0.0D0) THEN
        FUNC1=0.0D0
      ELSE
        FUNC1=-2.0D0*GX(TFF-X**2)*DEXP(-R1(NF)**2/4.0D0/ALPHA/X/X)
      END IF
      RETURN
      END

```

C

C*****C

```

      FUNCTION FUNC4(X)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER(MAXF=2500)
      COMMON /T/NF,INF,TMI,TMF,DELT,TRTM
      COMMON /C/ALPHA
      COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
      COMMON /R2/R2(MAXF),RV2(MAXF),MF,MT
      TFF=TMI+NF*DELT
      IF(X.EQ. 0.0D0) THEN
        FUNC4=0.0D0
      ELSE
        FUNC4=-2.0D0*GX(TFF-X**2)*DEXP(-R2(NF)**2/4.0D0/ALPHA/X/X)
      END IF

```

```

RETURN
END

```

C

C*****C

```

FUNCTION FUNC7(X)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
PARAMETER(MAXF=2500)
COMMON /T/NF,INF,TMI,TMF,DELTMT,TRTM
COMMON /C/ALPHA
COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
COMMON /R2/R2(MAXF),RV2(MAXF),MF,MT
TFF=TMI+NF*DELTMT
FUNC7=GX(X)*DEXP(-R2(NF)**2/4.0D0/ALPHA/(TFF-X))
$ /DSQRT(TFF-X)**3
RETURN
END

```

C

C*****C

```

FUNCTION FUNC12(X)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
PARAMETER(MAXF=2500)
COMMON /T/NF,INF,TMI,TMF,DELTMT,TRTM
COMMON /C/ALPHA
COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
TFF=TMI+NF*DELTMT
FUNC12=-2.0D0*GX(TFF-X**2)
RETURN
END

```

C

C*****C

```

FUNCTION FUNC8(X)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
PARAMETER (MAXF=2500)
DATA PI/3.14159265359D0/
COMMON/C/ALPHA
COMMON/T/NF,INF,TMI,TMF,DELTMT,TRTM
COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
FUNC8=DEXP(-X*X)
RETURN
END

```

C*****C

```

FUNCTION FUNC11(X)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
PARAMETER(MAXF=2500)
COMMON /T/NF,INF,TMI,TMF,DELTMT,TRTM
COMMON /C/ALPHA
COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
TFF=TMI+NF*DELTMT
IF(X .EQ. 0.0D0) THEN
  FUNC11=0.0D0
ELSE
  FUNC11=-2.0D0*DEXP(-R1(NF)**2/4.0D0/ALPHA/X/X)
END IF
RETURN

```

END

C
C*****C

```

FUNCTION FUNC44(X)
  IMPLICIT DOUBLE PRECISION(A-H,O-Z)
  PARAMETER(MAXF=2500)
  COMMON /T/NF,INF,TMI,TMF,DELTM,TRTM
  COMMON /C/ALPHA
  COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
  COMMON /R2/R2(MAXF),RV2(MAXF),MF,MT
  TFF=TMI+NF*DELTM
  IF(X .EQ. 0.0D0) THEN
    FUNC44=0.0D0
  ELSE
    FUNC44=-2.0D0*DEXP(-R2(NF)**2/4.0D0/ALPHA/X/X)
  END IF
  RETURN
END

```

C
C*****C

```

FUNCTION FUNC2A(X)
  IMPLICIT DOUBLE PRECISION(A-H,O-Z)
  PARAMETER(MAXF=2500)
  COMMON /T/NF,INF,TMI,TMF,DELTM,TRTM
  COMMON /C/ALPHA
  COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)

```

C

```

  TFF=TMI+NF*DELTM
  TF2=TMI+INF*DELTM
  TF1=TMI+(INF-1)*DELTM
  IF(INF .EQ. 1) THEN
    A=RV1(INF)
    B=-R1(INF)*TF1/(TF2-TF1)
  ELSE
    A=RV1(INF)
    B=(R1(INF-1)*TF2-R1(INF)*TF1)/(TF2-TF1)
  END IF
  RX=A*(TFF-X**2)+B

```

C

```

  IF(X .EQ. 0.0D0) THEN
    FUNC2A=0.0D0
  ELSE
    FUNC2A=-2.0D0*DEXP(-(R1(NF)+RX)**2/4.0D0/ALPHA/X/X)
  END IF
  RETURN
END

```

C*****C

```

FUNCTION FUNC5A(X)
  IMPLICIT DOUBLE PRECISION(A-H,O-Z)
  PARAMETER(MAXF=2500)
  COMMON /T/NF,INF,TMI,TMF,DELTM,TRTM
  COMMON /C/ALPHA
  COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
  COMMON /R2/R2(MAXF),RV2(MAXF),MF,MT

```

```

C
TFF=TMI+NF*DELTM
TF2=TMI+INF*DELTM
TF1=TMI+(INF-1)*DELTM
IF(INF .EQ. 1) THEN
  A=RV1(INF)
  B=-R1(INF)*TF1/(TF2-TF1)
ELSE
  A=RV1(INF)
  B=(R1(INF-1)*TF2-R1(INF)*TF1)/(TF2-TF1)
END IF
RX=A*(TFF-X**2)+B

C
IF(X .EQ. 0.0D0) THEN
  FUNC5A=0.0D0
ELSE
  FUNC5A=-2.0D0*DEXP(-(R2(NF)+RX)**2/4.0D0/ALPHA/X/X)
END IF
RETURN
END

C*****C
FUNCTION FUNC6A(X)
IMPLICIT DOUBLE PRECISION(A-H,0-Z)
PARAMETER(MAXF=2500)
COMMON /T/NF, INF, TMI, TMF, DELTM, TRTM
COMMON /C/ALPHA
COMMON /R1/R1(MAXF), RV1(MAXF), Q(MAXF)
COMMON /R2/R2(MAXF), RV2(MAXF), MF, MT

C
TFF=TMI+NF*DELTM
TF2=TMI+INF*DELTM
TF1=TMI+(INF-1)*DELTM
IF(INF .EQ. MT+1) THEN
  A=RV2(INF)
  B=-R2(INF)*TF1/(TF2-TF1)
ELSE
  A=RV2(INF)
  B=(R2(INF-1)*TF2-R2(INF)*TF1)/(TF2-TF1)
END IF
RX=A*(TFF-X**2)+B

C
IF(X .EQ. 0.0D0) THEN
  FUNC6A=0.0D0
ELSE
  FUNC6A=-2.0D0*DEXP(-(R2(NF)+RX)**2/4.0D0/ALPHA/X/X)
END IF
RETURN
END

C*****C
FUNCTION FUNC6B(X)
IMPLICIT DOUBLE PRECISION(A-H,0-Z)
PARAMETER(MAXF=2500)

```

```

COMMON /T/NF, INF, TMI, TMF, DELTM, TRTM
COMMON /C/ALPHA
COMMON /R1/R1(MAXF), RV1(MAXF), Q(MAXF)
COMMON /R2/R2(MAXF), RV2(MAXF), MF, MT

```

C

```

TFF=TMI+NF*DELTMT
TF2=TMI+INF*DELTMT
TF1=TMI+(INF-1)*DELTMT
IF(INF.EQ. MT+1) THEN
  A=RV2(INF)
  B=-R2(INF)*TF1/(TF2-TF1)
ELSE
  A=RV2(INF)
  B=(R2(INF-1)*TF2-R2(INF)*TF1)/(TF2-TF1)
END IF
RX=A*(TFF-X**2)+B

```

C

```

IF(X.EQ. 0.0D0) THEN
  FUNC6B=-2.0D0
ELSE
  FUNC6B=-2.0D0*DEXP(-(R2(NF)-RX)**2/4.0D0/ALPHA/X/X)
END IF
RETURN
END

```

C*****C

```

FUNCTION FUNC22A(X)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
PARAMETER(MAXF=2500)
COMMON /T/NF, INF, TMI, TMF, DELTM, TRTM
COMMON /C/ALPHA
COMMON /R1/R1(MAXF), RV1(MAXF), Q(MAXF)

```

C

```

TFF=TMI+NF*DELTMT
TF2=TMI+INF*DELTMT
TF1=TMI+(INF-1)*DELTMT
IF(INF.EQ. 1) THEN
  A=RV1(INF)
  B=-R1(INF)*TF1/(TF2-TF1)
ELSE
  A=RV1(INF)
  B=(R1(INF-1)*TF2-R1(INF)*TF1)/(TF2-TF1)
END IF
RX=A*(TFF-X**2)+B

```

C

```

IF(X.EQ. 0.0D0) THEN
  FUNC22A=0.0D0
ELSE
  FUNC22A=-4.0D0*DEXP(-RX**2/4.0D0/ALPHA/X/X)
END IF
RETURN
END

```

C

```

C*****C
  FUNCTION FUNC2B(X)
  IMPLICIT DOUBLE PRECISION(A-H,O-Z)
  PARAMETER(MAXF=2500)
  COMMON /T/NF,INF,TMI,TMF,DELTM,TRTM
  COMMON /C/ALPHA
  COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
C
  TFF=TMI+NF*DELTM
  TF2=TMI+INF*DELTM
  TF1=TMI+(INF-1)*DELTM
  IF(INF.EQ. 1) THEN
    A=RV1(INF)
    B=-R1(INF)*TF1/(TF2-TF1)
C
    B=0.0D0
  ELSE
    A=RV1(INF)
    B=(R1(INF-1)*TF2-R1(INF)*TF1)/(TF2-TF1)
  END IF
  RX=A*(TFF-X**2)+B
C
  IF(X.EQ. 0.0D0) THEN
    FUNC2B=-2.0D0
  ELSE
    FUNC2B=-2.0D0*DEXP(-(R1(NF)-RX)**2/4.0D0/ALPHA/X/X)
  END IF
  RETURN
  END
C
C*****C
  FUNCTION FUNC5B(X)
  IMPLICIT DOUBLE PRECISION(A-H,O-Z)
  PARAMETER(MAXF=2500)
  COMMON /T/NF,INF,TMI,TMF,DELTM,TRTM
  COMMON /C/ALPHA
  COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
  COMMON /R2/R2(MAXF),RV2(MAXF),MF,MT
C
  TFF=TMI+NF*DELTM
  TF2=TMI+INF*DELTM
  TF1=TMI+(INF-1)*DELTM
  IF(INF.EQ. 1) THEN
    A=RV1(INF)
    B=-R1(INF)*TF1/(TF2-TF1)
  ELSE
    A=RV1(INF)
    B=(R1(INF-1)*TF2-R1(INF)*TF1)/(TF2-TF1)
  END IF
  RX=A*(TFF-X**2)+B
C
  IF(X.EQ. 0.0D0) THEN
    IF(R1(NF).EQ. R2(NF)) THEN
      FUNC5B=-2.0D0
    ELSE

```



```

      FUNC5B=0.0D0
    ENDIF
  ELSE
    FUNC5B=-2.0D0*DEXP(-(R2(NF)-RX)**2/4.0D0/ALPHA/X/X)
  END IF
  RETURN
END

```

C
C*****C

```

  FUNCTION FUNC3A(X)
    IMPLICIT DOUBLE PRECISION(A-H,O-Z)
    PARAMETER(MAXF=2500)
    COMMON /T/NF,INF,TMI,TMF,DELTMT,TRTM
    COMMON /C/ALPHA
    COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
    COMMON /R2/R2(MAXF),RV2(MAXF),MF,MT

```

C

```

  TFF=TMI+NF*DELTMT
  TF2=TMI+INF*DELTMT
  TF1=TMI+(INF-1)*DELTMT
  IF(INF .EQ. MT+1) THEN
    A=RV2(INF)
    B=-R2(INF)*TF1/(TF2-TF1)
  ELSE
    A=RV2(INF)
    B=(R2(INF-1)*TF2-R2(INF)*TF1)/(TF2-TF1)
  END IF
  RX=A*(TFF-X**2)+B

```

C

```

  IF(X .EQ. 0.0D0) THEN
    FUNC3A=0.0D0
  ELSE
    FUNC3A=-2.0D0*DEXP(-(R1(NF)+RX)**2/4.0D0/ALPHA/X/X)
  END IF
  RETURN
END

```

C

C*****C

```

  FUNCTION FUNC3B(X)
    IMPLICIT DOUBLE PRECISION(A-H,O-Z)
    PARAMETER(MAXF=2500)
    COMMON /T/NF,INF,TMI,TMF,DELTMT,TRTM
    COMMON /C/ALPHA
    COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
    COMMON /R2/R2(MAXF),RV2(MAXF),MF,MT

```

C

```

  TFF=TMI+NF*DELTMT
  TF2=TMI+INF*DELTMT
  TF1=TMI+(INF-1)*DELTMT
  IF(INF .EQ. MT+1) THEN
    A=RV2(INF)
    B=-R2(INF)*TF1/(TF2-TF1)
  ELSE
    A=RV2(INF)

```



```

      B=(R2(INF-1)*TF2-R2(INF)*TF1)/(TF2-TF1)
END IF
RX=A*(TFF-X**2)+B
C
IF(X .EQ. 0.0D0) THEN
  IF(R1(NF) .EQ. R2(NF)) THEN
    FUNC3B=-2.0D0
  ELSE
    FUNC3B=0.0D0
  END IF
ELSE
  FUNC3B=-2.0D0*DEXP(-(R1(NF)-RX)**2/4.0D0/ALPHA/X/X)
END IF
RETURN
END

C
C*****C
C  SUBPROGRAM BRENT'S
C  TO FIND THE ROOT OF FUNCTION TFUNCA
C*****C
FUNCTION ZBRENT(TFUNC,X1,X2,TOL)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
EXTERNAL TFUNC
PARAMETER (ITMAX=100,EPS=3.D-16)
A=X1
B=X2
FA=TFUNC(A)
FB=TFUNC(B)
IF(FB*FA.GT.0.0D0) PAUSE 'ROOT MUST BE BRACKETED FOR ZBRENT.'
FC=FB
DO 11 ITER=1,ITMAX
  IF(FB*FC.GT.0.0D0) THEN
    C=A
    FC=FA
    D=B-A
    E=D
  ENDIF
  IF(DABS(FC).LT.DABS(FB)) THEN
    A=B
    B=C
    C=A
    FA=FB
    FB=FC
    FC=FA
  ENDIF
  TOL1=2.0D0*EPS*DABS(B)+0.5D0*TOL
  XM=.5D0*(C-B)
  IF(DABS(XM).LE.TOL1 .OR. FB.EQ.0.0D0)THEN
    ZBRENT=B
    RETURN
  ENDIF
  IF(DABS(E).GE.TOL1 .AND. DABS(FA).GT.DABS(FB)) THEN
    S=FB/FA
    IF(A.EQ.C) THEN

```

```

        P=2.D0*XM*S
        Q=1.D0-S
    ELSE
        Q=FA/FC
        R=FB/FC
        P=S*(2.D0*XM*Q*(Q-R)-(B-A)*(R-1.D0))
        Q=(Q-1.D0)*(R-1.D0)*(S-1.D0)
    ENDIF
    IF(P.GT.0.D0) Q=-Q
    P=DABS(P)
    IF(2.D0*P .LT. DMIN1(3.*XM*Q-DABS(TOL1*Q),DABS(E*Q)))
$      THEN
        E=D
        D=P/Q
    ELSE
        D=XM
        E=D
    ENDIF
    ELSE
        D=XM
        E=D
    ENDIF
    A=B
    FA=FB
    IF(DABS(D) .GT. TOL1) THEN
        B=B+D
    ELSE
        B=B+DSIGN(TOL1, XM)
    ENDIF
    FB=TFUNC(B)
11  CONTINUE
    PAUSE 'ZBRENT EXCEEDING MAXIMUM ITERATIONS.'
    ZBRENT=B
    RETURN
    END
C*****C
C    ROMBERG INTEGRATION PROGRAM                                C
C    REFERRED TO HORNBECK'BOOK PP.154                          C
C    INPUT--A,B,EPS (INTERVAL AND CRITERION)                   C
C*****C
C    SUBROUTINE ROMBERG(FUNC,A,B,RESULT)
C    IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C    EXTERNAL FUNC
C    PARAMETER(MAX=40,EPS=0.0001D0)
C    DIMENSION T(MAX,MAX)
C
C    T(1,1)=(B-A)*(FUNC(A)+FUNC(B))/2.0D0
C    T(1,2)=T(1,1)/2.0D0+(B-A)*FUNC((A+B)/2.0D0)/2.0D0
C    T(2,1)=(4.0D0*T(1,2)-T(1,1))/3.0D0
C    J=3
C
C    C-----SUCCESSIVE APPLICATION OF TRAPEZOIDAL RULE
C
50  DELX=(B-A)/2.0D0**(J-1)

```

```

      X=A-DELX
      N=2** (J-2)
      SUM=0.0D0
      DO 100 I=1,N
          X=X+2.0D0*DELX
          SUM=SUM+FUNC(X)
100    CONTINUE
      T(1,J)=T(1,J-1)/2.0D0+DELX*SUM
C
C-----EXTRAPOLATION
C
      DO 200 L=2,J
          K=J+1-L
          T(L,K)=(4.0D0** (L-1)*T(L-1,K+1)-T(L-1,K))/
$      (4.0D0** (L-1)-1.0D0)
200    CONTINUE
C
C-----CHECK ACCURACY CRITERION
C
      IF(T(J,1) .EQ. 0.0D0) THEN
          RESULT=T(J,1)
          GO TO 111
      END IF
C
      IF(DABS((T(J,1)-T(J-1,1))/T(J,1)) .GE. EPS) THEN
          J=J+1
          IF(J.GT.MAX) THEN
              PAUSE 'TOO MANY STEPS'
              GO TO 111
          ELSE
              GO TO 50
          END IF
      ELSE
          RESULT=T(J,1)
      END IF
111    RETURN
      END
C*****C
      FUNCTION GX(X)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      GX=0.5D0
      END
C*****C
      FUNCTION ERF(X)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      IF(X.LT.0.0D0)THEN
          ERF=-GAMMP(.5D0,X**2)
      ELSE
          ERF=GAMMP(.5D0,X**2)
      ENDIF
      RETURN
      END
C

```

```

C*****C
FUNCTION ERFC(X)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
IF(X.LT.0.0D0)THEN
  ERFC=1.0D0+GAMMP(.5D0,X**2)
ELSE
  ERFC=GAMMQ(.5D0,X**2)
ENDIF
RETURN
END

C*****C
FUNCTION GAMMP(A,X)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
IF(X.LT.0.0D0 .OR. A.LE.0.0D0)PAUSE
IF(X.LT.A+1.0D0)THEN
  CALL GSER(GAMSER,A,X,GLN)
  GAMMP=GAMSER
ELSE
  CALL GCF(GAMMCF,A,X,GLN)
  GAMMP=1.0D0-GAMMCF
ENDIF
RETURN
END

C*****C
FUNCTION GAMMQ(A,X)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
IF(X.LT.0.0D0 .OR. A.LE.0.0D0)PAUSE
IF(X.LT.A+1.0D0)THEN
  CALL GSER(GAMSER,A,X,GLN)
  GAMMQ=1.0D0-GAMSER
ELSE
  CALL GCF(GAMMCF,A,X,GLN)
  GAMMQ=GAMMCF
ENDIF
RETURN
END

C*****C
SUBROUTINE GCF(GAMMCF,A,X,GLN)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
PARAMETER (ITMAX=200,EPS=3.D-12)
GLN=GAMMLN(A)
GOLD=0.0D0
A0=1.0D0
A1=X
B0=0.0D0
B1=1.0D0
FAC=1.0D0
DO 11 N=1,ITMAX
  AN=DFLOAT(N)
  ANA=AN-A
  A0=(A1+A0*ANA)*FAC
  B0=(B1+B0*ANA)*FAC
  ANF=AN*FAC
  A1=X*A0+ANF*A1

```

```

      B1=X*B0+ANF*B1
      IF(A1.NE.0.0D0)THEN
        FAC=1.0D0/A1
        G=B1*FAC
        IF(DABS((G-GOLD)/G).LT.EPS)GO TO 1
        GOLD=G
      ENDIF
11  CONTINUE
      PAUSE 'A TOO LARGE, ITMAX TOO SMALL'
1   GAMMCF=DEXP(-X+A*DLOG(X)-GLN)*G
      RETURN
      END
C*****C
      SUBROUTINE GSER(GAMSER,A,X,GLN)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER (ITMAX=200,EPS=3.D-12)
      GLN=GAMMLN(A)
      IF(X.LE.0.0D0)THEN
        IF(X.LT.0.0D0)PAUSE 'X LT 0.0D0'
        GAMSER=0.0D0
        RETURN
      ENDIF
      AP=A
      SUM=1.0D0/A
      DEL=SUM
      DO 11 N=1,ITMAX
        AP=AP+1.0D0
        DEL=DEL*X/AP
        SUM=SUM+DEL
        IF(DABS(DEL).LT.DABS(SUM)*EPS)GO TO 1
11  CONTINUE
      PAUSE 'A TOO LARGE, ITMAX TOO SMALL'
1   GAMSER=SUM*DEXP(-X+A*DLOG(X)-GLN)
      RETURN
      END
C*****C
      FUNCTION GAMMLN(XX)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION COF(6)
      DATA COF,STP/76.180091729406D0,-86.505320327112D0,
$      24.01409822230D0,-1.231739516140D0,.120858003D-2,
$      -.536382D-5,2.50662827465D0/
      DATA HALF,ONE,FPF/0.5D0,1.0D0,5.5D0/
      X=XX-ONE
      TMP=X+FPF
      TMP=(X+HALF)*DLOG(TMP)-TMP
      SER=ONE
      DO 11 J=1,6
        X=X+ONE
        SER=SER+COF(J)/X
11  CONTINUE
      GAMMLN=TMP+DLOG(STP*SER)
      RETURN
      END

```

```

C*****C
C  SUBPROGRAM LU DECOMPOSITION
C*****C
SUBROUTINE LUDCMP(A,N,NP,INDX,D)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
PARAMETER (NMAX=100,TINY=1.0D-20)
DIMENSION A(NP,NP),INDX(N),VV(NMAX)
D=1.0D0
DO 12 I=1,N
  AAMAX=0.0D0
  DO 11 J=1,N
    IF (DABS(A(I,J)).GT.AAMAX) AAMAX=DABS(A(I,J))
11  CONTINUE
    IF (AAMAX.EQ.0.0D0) PAUSE 'SINGULAR MATRIX.'
    VV(I)=1.0D0/AAMAX
12  CONTINUE
    DO 19 J=1,N
      IF (J.GT.1) THEN
        DO 14 I=1,J-1
          SUM=A(I,J)
          IF (I.GT.1) THEN
            DO 13 K=1,I-1
              SUM=SUM-A(I,K)*A(K,J)
13          CONTINUE
              A(I,J)=SUM
            ENDIF
          CONTINUE
14        ENDIF
          AAMAX=0.0D0
          DO 16 I=J,N
            SUM=A(I,J)
            IF (J.GT.1) THEN
              DO 15 K=1,J-1
                SUM=SUM-A(I,K)*A(K,J)
15          CONTINUE
                A(I,J)=SUM
              ENDIF
              DUM=VV(I)*DABS(SUM)
              IF (DUM.GE.AAMAX) THEN
                IMAX=I
                AAMAX=DUM
              ENDIF
16          CONTINUE
            IF (J.NE.IMAX) THEN
              DO 17 K=1,N
                DUM=A(IMAX,K)
                A(IMAX,K)=A(J,K)
                A(J,K)=DUM
17          CONTINUE
              D=-D
              VV(IMAX)=VV(J)
            ENDIF
            INDX(J)=IMAX
            IF (J.NE.N) THEN

```

```

      IF(A(J,J).EQ.0.0D0)A(J,J)=TINY
      DUM=1.0D0/A(J,J)
      DO 18 I=J+1,N
        A(I,J)=A(I,J)*DUM
18      CONTINUE
      ENDIF
19      CONTINUE
      IF(A(N,N).EQ.0.0D0)A(N,N)=TINY
      RETURN
      END

```

```

C*****C
C      SUBPROGRAM BACKSUBSTITUTION      C
C*****C
      SUBROUTINE LUBKSB(A,N,NP,INDX,B)
      IMPLICIT DOUBLE PRECISION(A-H,0-Z)
      DIMENSION A(NP,NP),INDX(N),B(N)
      II=0.0D0
      DO 12 I=1,N
        LL=INDX(I)
        SUM=B(LL)
        B(LL)=B(I)
        IF (II.NE.0.0D0)THEN
          DO 11 J=II,I-1
            SUM=SUM-A(I,J)*B(J)
11          CONTINUE
          ELSE IF (SUM.NE.0.0D0) THEN
            II=I
          ENDIF
        B(I)=SUM
12      CONTINUE
      DO 14 I=N,1,-1
        SUM=B(I)
        IF(I.LT.N)THEN
          DO 13 J=I+1,N
            SUM=SUM-A(I,J)*B(J)
13          CONTINUE
        ENDIF
        B(I)=SUM/A(I,I)
14      CONTINUE
      RETURN
      END
C*****C
      FUNCTION RAN3(IDUM)
      IMPLICIT DOUBLE PRECISION(A-H,0-Z)
C      PARAMETER (MBIG=4000000.,MSEED=1618033.,MZ=0.,FAC=2.5E-7)
      PARAMETER (MBIG=1000000000,MSEED=161803398,MZ=0,FAC=1.E-9)
      DIMENSION MA(55)
      DATA IFF /0/
      IF(IDUM.LT.0.OR.IFF.EQ.0)THEN
        IFF=1
        MJ=MSEED-IABS(IDUM)
        MJ=MOD(MJ,MBIG)
        MA(55)=MJ

```



```

      MK=1
      DO 11 I=1,54
        II=MOD(21*I,55)
        MA(II)=MK
        MK=MJ-MK
        IF(MK.LT.MZ)MK=MK+MBIG
        MJ=MA(II)
11      CONTINUE
      DO 13 K=1,4
        DO 12 I=1,55
          MA(I)=MA(I)-MA(1+MOD(I+30,55))
          IF(MA(I).LT.MZ)MA(I)=MA(I)+MBIG
12      CONTINUE
13      CONTINUE
      INEXT=0
      INEXTP=31
      IDUM=1
      ENDIF
      INEXT=INEXT+1
      IF(INEXT.EQ.56)INEXT=1
      INEXTP=INEXTP+1
      IF(INEXTP.EQ.56)INEXTP=1
      MJ=MA(INEXT)-MA(INEXTP)
      IF(MJ.LT.MZ)MJ=MJ+MBIG
      MA(INEXT)=MJ
      RAN3=MJ*FAC
      RETURN
      END
C*****C
      SUBROUTINE MPROVE(A,ALUD,N,NP,INDX,B,X)
      IMPLICIT DOUBLE PRECISION(A-H,0-Z)
      PARAMETER (NMAX=100)
      DIMENSION A(NP,NP),ALUD(NP,NP),INDX(N),B(N),X(N),R(NMAX)
      REAL*8 SDP
      DO 12 I=1,N
        SDP=-B(I)
        DO 11 J=1,N
          SDP=SDP+DBLE(A(I,J))*DBLE(X(J))
11      CONTINUE
        R(I)=SDP
12      CONTINUE
      CALL LUBKSB(ALUD,N,NP,INDX,R)
      DO 13 I=1,N
        X(I)=X(I)-R(I)
13      CONTINUE
      RETURN
      END
C*****C
C      SUBPROGRAM C
C      INTERNAL POINTS TEMPERATURE COMPUTATION C
C*****C
      SUBROUTINE INTERNAL(TINTL)
      IMPLICIT DOUBLE PRECISION(A-H,0-Z)
      PARAMETER(MAXF=2500,PI=3.141592653589793D0)

```

```

COMMON /T/NF, INF, TMI, TMF, DELTM, TRTM
COMMON /C/ALPHA
COMMON /C1/CK, RH0, CP, CLF, CLV, TM, TV
COMMON /R1/R1(MAXF), RV1(MAXF), Q(MAXF)
COMMON /R2/R2(MAXF), RV2(MAXF), MF, MT
COMMON /RINT/RIX
EXTERNAL FINTL1, FINL2A, FINL2B

```

C

```
TFF=TMI+NF*DELT
```

C

```

IF(TMI .EQ. 0.0D0)THEN
  QINT=0.0D0
ELSE
  AA=DSQRT(TFF)
  BB=DSQRT(TFF-TMI)
  CALL ROMBERG(FINTL1, AA, BB, QINT)
ENDIF

```

C

```

QINT1=0.0D0
DO INF=1, NF
  TF2=TMI+INF*DELT
  TF1=TMI+(INF-1)*DELT
  AA=DSQRT(TFF-TF1)
  BB=DSQRT(TFF-TF2)
  CALL ROMBERG(FINTL1, AA, BB, A)
  QINT1=QINT1+A
ENDDO

```

C

```

SUMB1=0.0D0
SUMB2=0.0D0
DO INF=1, NF
  TF2=TMI+INF*DELT
  TF1=TMI+(INF-1)*DELT
  AA=DSQRT(TFF-TF1)
  BB=DSQRT(TFF-TF2)
  CALL ROMBERG(FINL2A, AA, BB, B1)
  CALL ROMBERG(FINL2B, AA, BB, B2)
  B1=RV1(INF)*B1
  B2=RV1(INF)*B2
  SUMB1=SUMB1+B1
  SUMB2=SUMB2+B2
ENDDO
SUMB=SUMB1+SUMB2

```

C

```

TINTL=(2.0D0*ALPHA/CK*(QINT+QINT1)-CLF/CP*SUMB)
$ /2.0D0/DSQRT(PI*ALPHA)
TIME=TMI+DELT*NF
WRITE(*,*)TIME, RIX, TINTL
WRITE(77,555)TIME, RIX, TINTL
WRITE(71,*)RIX
WRITE(90,*)TIME
WRITE(72,*)TINTL
555 FORMAT(2X,F10.7,5X,E14.7,5X,E14.7)
RETURN

```

END

```

C
C*****C
  FUNCTION FINTL1(X)
  IMPLICIT DOUBLE PRECISION(A-H,O-Z)
  PARAMETER(MAXF=2500)
  COMMON /T/NF,INF,TMI,TMF,DELTM,TRTM
  COMMON /C/ALPHA
  COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
  COMMON /RINT/RIX

C
  TFF=TMI+NF*DELTM
  IF(X .EQ. 0.0D0) THEN
    FINTL1=0.0D0
  ELSE
    FINTL1=-2.0D0*GX(TFF-X**2)*DEXP(-RIX**2/4.0D0/ALPHA/X/X)
  END IF
  RETURN
  END

C
C*****C
  FUNCTION FINL2A(X)
  IMPLICIT DOUBLE PRECISION(A-H,O-Z)
  PARAMETER(MAXF=2500)
  COMMON /T/NF,INF,TMI,TMF,DELTM,TRTM
  COMMON /C/ALPHA
  COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
  COMMON /RINT/RIX

C
  TFF=TMI+NF*DELTM
  TF2=TMI+INF*DELTM
  TF1=TMI+(INF-1)*DELTM
  IF(INF .EQ. 1) THEN
    A=RV1(INF)
    B=-R1(INF)*TF1/(TF2-TF1)
  ELSE
    A=RV1(INF)
    B=(R1(INF-1)*TF2-R1(INF)*TF1)/(TF2-TF1)
  END IF
  RX=A*(TFF-X**2)+B

C
  IF(X .EQ. 0.0D0) THEN
    FINL2A=0.0D0
  ELSE
    FINL2A=-2.0D0*DEXP(-(RIX+RX)**2/4.0D0/ALPHA/X/X)
  END IF
  RETURN
  END

C*****C
  FUNCTION FINL2B(X)
  IMPLICIT DOUBLE PRECISION(A-H,O-Z)
  PARAMETER(MAXF=2500)
  COMMON /T/NF,INF,TMI,TMF,DELTM,TRTM
  COMMON /C/ALPHA

```

```

COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
COMMON /RINT/RIX
C
TFF=TMI+NF*DELTM
TF2=TMI+INF*DELTM
TF1=TMI+(INF-1)*DELTM
IF(INF .EQ. 1) THEN
  A=RV1(INF)
  B=-R1(INF)*TF1/(TF2-TF1)
ELSE
  A=RV1(INF)
  B=(R1(INF-1)*TF2-R1(INF)*TF1)/(TF2-TF1)
END IF
RX=A*(TFF-X**2)+B
C
IF(X .EQ. 0.0D0) THEN
  IF(RIX .EQ. R1(NF)) THEN
    FINL2B=-2.0D0
  ELSE
    FINL2B=0.0D0
  ENDIF
ELSE
  FINL2B=-2.0D0*DEXP(-(RIX-RX)**2/4.0D0/ALPHA/X/X)
END IF
RETURN
END
C*****C
C SUBPROGRAM C
C STORED HEAT COMPUTATION C
C*****C
SUBROUTINE HEATS
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
PARAMETER(MAXF=2500,PI=3.141592653589793D0)
COMMON /T/NF,INF,TMI,TMF,DELTM,TRTM
COMMON /C1/CK,RH0,CP,CLF,CLV,TM,TV
COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
COMMON /R2/R2(MAXF),RV2(MAXF),MF,MT
COMMON /HEAT1/RX(300),TIN(300)
C
TFF=TMI+NF*DELTM
C
C-----COMPUTE THE STORED LATENT HEAT IN THE MATERIAL
C
SLH=RH0*(R1(NF)-R2(NF))*CLF+RH0*R2(NF)*(CLV+CLF+CP*TV)
C
C-----COMPUTE THE STORED SENSIBLE HEAT IN THE MATERIAL
C (SIMPSON'S 1/3 RULE)
C
IF(NF .LE. MT) THEN
  SSH1=(RX(2)-RX(1))/3.0D0*(TIN(1)+4.0D0*TIN(2)+
$ 2.0D0*TIN(3)+4.0D0*TIN(4)+2.0D0*TIN(5)+4.0D0*TIN(6)+
$ 2.0D0*TIN(7)+4.0D0*TIN(8)+2.0D0*TIN(9)+4.0D0*TIN(10)+
$ TIN(11))
  SSH2=(RX(12)-RX(11))/3.0D0*(TIN(11)+4.0D0*TIN(12)+

```

```

$      2.0D0*TIN(13)+4.0D0*TIN(14)+2.0D0*TIN(15)+4.0D0*TIN(16)+
$      2.0D0*TIN(17)+4.0D0*TIN(18)+2.0D0*TIN(19)+4.0D0*TIN(20)+
$      2.0D0*TIN(21)+4.0D0*TIN(22)+2.0D0*TIN(23)+4.0D0*TIN(24)+
$      2.0D0*TIN(25)+4.0D0*TIN(26)+2.0D0*TIN(27)+4.0D0*TIN(28)+
$      2.0D0*TIN(29)+4.0D0*TIN(30)+2.0D0*TIN(31)+4.0D0*TIN(32)+
$      2.0D0*TIN(33)+4.0D0*TIN(34)+2.0D0*TIN(35)+4.0D0*TIN(36)+
$      2.0D0*TIN(37)+4.0D0*TIN(38)+2.0D0*TIN(39)+4.0D0*TIN(40)+
$      2.0D0*TIN(41)+4.0D0*TIN(42)+2.0D0*TIN(43)+4.0D0*TIN(44)+
$      2.0D0*TIN(45)+4.0D0*TIN(46)+2.0D0*TIN(47)+4.0D0*TIN(48)+
$      2.0D0*TIN(49)+4.0D0*TIN(50)+2.0D0*TIN(51)+4.0D0*TIN(52)+
$      TIN(53))
      SSH=SSH1+SSH2
      SSH=RHO*CP*SSH
ELSE
      SSH1=0.0D0
      SSH2=(RX(12)-RX(11))/3.0D0*(TIN(11)+4.0D0*TIN(12)+
$      2.0D0*TIN(13)+4.0D0*TIN(14)+2.0D0*TIN(15)+4.0D0*TIN(16)+
$      2.0D0*TIN(17)+4.0D0*TIN(18)+2.0D0*TIN(19)+4.0D0*TIN(20)+
$      2.0D0*TIN(21)+4.0D0*TIN(22)+2.0D0*TIN(23)+4.0D0*TIN(24)+
$      2.0D0*TIN(25)+4.0D0*TIN(26)+2.0D0*TIN(27)+4.0D0*TIN(28)+
$      2.0D0*TIN(29)+4.0D0*TIN(30)+2.0D0*TIN(31)+4.0D0*TIN(32)+
$      2.0D0*TIN(33)+4.0D0*TIN(34)+2.0D0*TIN(35)+4.0D0*TIN(36)+
$      2.0D0*TIN(37)+4.0D0*TIN(38)+2.0D0*TIN(39)+4.0D0*TIN(40)+
$      2.0D0*TIN(41)+4.0D0*TIN(42)+2.0D0*TIN(43)+4.0D0*TIN(44)+
$      2.0D0*TIN(45)+4.0D0*TIN(46)+2.0D0*TIN(47)+4.0D0*TIN(48)+
$      2.0D0*TIN(49)+4.0D0*TIN(50)+2.0D0*TIN(51)+4.0D0*TIN(52)+
$      2.0D0*TIN(53)+4.0D0*TIN(54)+2.0D0*TIN(55)+4.0D0*TIN(56)+
$      2.0D0*TIN(57)+4.0D0*TIN(58)+2.0D0*TIN(59)+4.0D0*TIN(60)+
$      TIN(61))
      SSH3=(RX(62)-RX(61))/3.0D0*(TIN(61)+4.0D0*TIN(62)+
$      2.0D0*TIN(63)+4.0D0*TIN(64)+2.0D0*TIN(65)+4.0D0*TIN(66)+
$      2.0D0*TIN(67)+4.0D0*TIN(68)+2.0D0*TIN(69)+4.0D0*TIN(70)+
$      2.0D0*TIN(71)+4.0D0*TIN(72)+2.0D0*TIN(73)+4.0D0*TIN(74)+
$      2.0D0*TIN(75)+4.0D0*TIN(76)+2.0D0*TIN(77)+4.0D0*TIN(78)+
$      2.0D0*TIN(79)+4.0D0*TIN(80)+2.0D0*TIN(81)+4.0D0*TIN(82)+
$      2.0D0*TIN(83)+4.0D0*TIN(84)+2.0D0*TIN(85)+4.0D0*TIN(86)+
$      2.0D0*TIN(87)+4.0D0*TIN(88)+2.0D0*TIN(89)+4.0D0*TIN(90)+
$      TIN(91))
      SSH=SSH1+SSH2+SSH3
      SSH=RHO*CP*SSH
END IF
TOTAL=SLH+SSH
WRITE(77,200)TOTAL,SLH,SSH
200  FORMAT(/1X,'STORED HEAT=',E14.7,2X,'LATENT=',E14.7,2X
$, 'SENSIBLE=',E14.7)
      RETURN
      END
C*****C
C      SUBPROGRAM                      C
C      INTERNAL POINTS TEMPERATURE COMPUTATION          C
C*****C
      SUBROUTINE INTERNAL1(TINTL)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER(MAXF=2500,PI=3.141592653589793D0)

```



```

COMMON /T/NF, INF, TMI, TMF, DELTM, TRTM
COMMON /C/ALPHA
COMMON /C1/CK, RHO, CP, CLF, CLV, TM, TV
COMMON /R1/R1(MAXF), RV1(MAXF), Q(MAXF)
COMMON /R2/R2(MAXF), RV2(MAXF), MF, MT
COMMON /RINT/RIX
EXTERNAL FINTL1, FINL2A, FINL2B, FINTL3, FINTL4, FINTL5A,
$      FINTL5B, FINTL6B, FINTL6A

```

C

```
TFF=TMI+NF*DELT
```

C

```

AA=DSQRT(TFF)
BB=DSQRT(TFF-TRTM)
CALL ROMBERG(FINTL3, AA, BB, QINTT)

```

C

```

SUMA=0.0D0
DO INF=MT+1, NF
    TF2=TMI+INF*DELT
    TF1=TMI+(INF-1)*DELT
    AA=DSQRT(TFF-TF1)
    BB=DSQRT(TFF-TF2)
    CALL ROMBERG(FINTL4, AA, BB, A)
    A=A*Q(INF)
    SUMA=SUMA+A
ENDDO

```

C

```

SUMB1=0.0D0
SUMB2=0.0D0
DO INF=1, NF
    TF2=TMI+INF*DELT
    TF1=TMI+(INF-1)*DELT
    AA=DSQRT(TFF-TF1)
    BB=DSQRT(TFF-TF2)
    CALL ROMBERG(FINTL5A, AA, BB, B1)
    CALL ROMBERG(FINTL5B, AA, BB, B2)
    B1=RV1(INF)*B1
    B2=RV1(INF)*B2
    SUMB1=SUMB1+B1
    SUMB2=SUMB2+B2
ENDDO
SUMB=SUMB1+SUMB2

```

C

```

SUMC1=0.0D0
SUMC2=0.0D0
DO INF=MT+1, NF
    TF2=TMI+INF*DELT
    TF1=TMI+(INF-1)*DELT
    AA=DSQRT(TFF-TF1)
    BB=DSQRT(TFF-TF2)
    CALL ROMBERG(FINTL6A, AA, BB, C1)
    CALL ROMBERG(FINTL6B, AA, BB, C2)
    C1=RV2(INF)*C1
    C2=RV2(INF)*C2
    SUMC1=SUMC1+C1

```

```

      SUMC2=SUMC2+C2
ENDDO
SUMC=SUMC1+SUMC2
C
  TINTL=DSQRT( ALPHA/PI )/CK*(QINTT+SUMA)-1.0D0/2.0D0/CP/
$    DSQRT(PI*ALPHA)*(CLF*SUMB+CLV*SUMC)
  TIME=TMI+DELTN*NF
  WRITE(*,*)TIME,RIX,TINTL
  WRITE(77,556)TIME,RIX,TINTL
  WRITE(90,*)TIME
  WRITE(71,*)RIX
  WRITE(72,*)TINTL+300.D0
556  FORMAT(2X,F10.7,5X,E14.7,5X,E14.7)
  RETURN
  END
C*****C
  FUNCTION FINTL3(X)
  IMPLICIT DOUBLE PRECISION(A-H,0-Z)
  PARAMETER(MAXF=2500)
  COMMON /T/NF,INF,TMI,TMF,DELTN,TRTM
  COMMON /C/ALPHA
  COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
  COMMON /RINT/RIX
C
  TFF=TMI+NF*DELTN
  IF(X.EQ. 0.0D0) THEN
    FINTL3=0.0D0
  ELSE
    FINTL3=-2.0D0*GX(TFF-X**2)*DEXP(-RIX**2/4.0D0/ALPHA/X/X)
  END IF
  RETURN
  END
C
C*****C
  FUNCTION FINTL4(X)
  IMPLICIT DOUBLE PRECISION(A-H,0-Z)
  PARAMETER(MAXF=2500)
  COMMON /T/NF,INF,TMI,TMF,DELTN,TRTM
  COMMON /C/ALPHA
  COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
  COMMON /RINT/RIX
C
  TFF=TMI+NF*DELTN
  IF(X.EQ. 0.0D0) THEN
    FINTL4=0.0D0
  ELSE
    FINTL4=-2.0D0*DEXP(-RIX**2/4.0D0/ALPHA/X/X)
  END IF
  RETURN
  END
C
C*****C
  FUNCTION FINTL5A(X)
  IMPLICIT DOUBLE PRECISION(A-H,0-Z)

```



```

PARAMETER(MAXF=2500)
COMMON /T/NF, INF, TMI, TMF, DELTM, TRTM
COMMON /C/ALPHA
COMMON /R1/R1(MAXF), RV1(MAXF), Q(MAXF)
COMMON /RINT/RIX

```

```

C
TFF=TMI+NF*DELT
TF2=TMI+INF*DELT
TF1=TMI+(INF-1)*DELT
IF(INF .EQ. 1) THEN
  A=RV1(INF)
  B=-R1(INF)*TF1/(TF2-TF1)
ELSE
  A=RV1(INF)
  B=(R1(INF-1)*TF2-R1(INF)*TF1)/(TF2-TF1)
END IF
RX=A*(TFF-X**2)+B

```

```

C
IF(X .EQ. 0.0D0) THEN
  FINTL5A=0.0D0
ELSE
  FINTL5A=-2.0D0*DEXP(-(RIX+RX)**2/4.0D0/ALPHA/X/X)
END IF
RETURN
END

```

```

C
C*****C
FUNCTION FINTL5B(X)
  IMPLICIT DOUBLE PRECISION(A-H,O-Z)
  PARAMETER(MAXF=2500)
  COMMON /T/NF, INF, TMI, TMF, DELTM, TRTM
  COMMON /C/ALPHA
  COMMON /R1/R1(MAXF), RV1(MAXF), Q(MAXF)
  COMMON /R2/R2(MAXF), RV2(MAXF), MF, MT
  COMMON /RINT/RIX

```

```

C
TFF=TMI+NF*DELT
TF2=TMI+INF*DELT
TF1=TMI+(INF-1)*DELT
IF(INF .EQ. 1) THEN
  A=RV1(INF)
  B=-R1(INF)*TF1/(TF2-TF1)
ELSE
  A=RV1(INF)
  B=(R1(INF-1)*TF2-R1(INF)*TF1)/(TF2-TF1)
END IF
RX=A*(TFF-X**2)+B

```

```

C
IF(X .EQ. 0.0D0) THEN
  IF(RIX .EQ. R1(NF)) THEN
    FINTL5B=-2.0D0
  ELSE
    FINTL5B=0.0D0
  ENDIF

```

```

ELSE
  FINTL5B=-2.0D0*DEXP(-(RIX-RX)**2/4.0D0/ALPHA/X/X)
END IF
RETURN
END

```

```

C
C*****C

```

```

FUNCTION FINTL6A(X)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
PARAMETER(MAXF=2500)
COMMON /T/NF,INF,TMI,TMF,DELTM,TRTM
COMMON /C/ALPHA
COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
COMMON /R2/R2(MAXF),RV2(MAXF),MF,MT
COMMON /RINT/RIX

```

```

C
  TFF=TMI+NF*DELTM
  TF2=TMI+INF*DELTM
  TF1=TMI+(INF-1)*DELTM
  IF(INF.EQ.MT+1) THEN
    A=RV2(INF)
    B=-R2(INF)*TF1/(TF2-TF1)
  ELSE
    A=RV2(INF)
    B=(R2(INF-1)*TF2-R2(INF)*TF1)/(TF2-TF1)
  END IF
  RX=A*(TFF-X**2)+B

```

```

C
  IF(X.EQ.0.0D0) THEN
    FINTL6A=0.0D0
  ELSE
    FINTL6A=-2.0D0*DEXP(-(RIX+RX)**2/4.0D0/ALPHA/X/X)
  END IF
  RETURN
END

```

```

C
C*****C

```

```

FUNCTION FINTL6B(X)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
PARAMETER(MAXF=2500)
COMMON /T/NF,INF,TMI,TMF,DELTM,TRTM
COMMON /C/ALPHA
COMMON /R1/R1(MAXF),RV1(MAXF),Q(MAXF)
COMMON /R2/R2(MAXF),RV2(MAXF),MF,MT
COMMON /RINT/RIX

```

```

C
  TFF=TMI+NF*DELTM
  TF2=TMI+INF*DELTM
  TF1=TMI+(INF-1)*DELTM
  IF(INF.EQ.MT+1) THEN
    A=RV2(INF)
    B=-R2(INF)*TF1/(TF2-TF1)
  ELSE
    A=RV2(INF)

```

C

```

      B=(R2(INF-1)*TF2-R2(INF)*TF1)/(TF2-TF1)
END IF
RX=A*(TFF-X**2)+B

IF(X .EQ. 0.0D0) THEN
  IF(RIX .EQ. R2(NF)) THEN
    FINTL6B=-2.0D0
  ELSE
    FINTL6B=0.0D0
  END IF
ELSE
  FINTL6B=-2.0D0*DEXP(-(RIX-RX)**2/4.0D0/ALPHA/X/X)
END IF
RETURN
END

```

REFERENCES

1. Zabaras, N., Mukherjee, S., Richmond, O., 1988, "An analysis of inverse heat transfer problems with phase change using an integral method," *J. Heat Transfer*, Vol. 110, pp. 554-561.
2. Hsieh, C. K., Choi, C. Y., 1992, "Solution of one- and two-phase melting and solidification problems imposed with constant or time-variant temperature and flux boundary conditions," *J. Heat Transfer*, Vol. 114, pp. 524-528.
3. Hsieh, C. K., Akbari, M., and Li, H., 1992, "Solution of inverse Stefan problems by a source-and-sink method," *Int. J. Num. Meth. Heat Fluid Flow*, Vol. 2, pp. 391-406.
4. Choi, C. Y., Hsieh, C. K., 1992, "Solution of Stefan problems imposed with cyclic temperature and flux boundary conditions," *Int. J. Heat Mass Transfer*, Vol. 35, pp. 1181-1195.
5. Choi, C. Y., 1991, "Exact and numerical solution of one- and two-phase melting and solidification problems imposed with constant or time-variant temperature and flux conditions," PhD Dissertation, Univ. of Florida.
6. Rubinstein, L. I., 1971, *The Stefan problem*, Am. Math. Soc. Transl. Math. Monogr., No. 27.
7. Carslaw, H. S., and Jaeger, J. C., 1959, *Conduction of Heat in Solids*, 2nd ed., Oxford Univ. Press, London and New York.
8. Muehlbauer, J. C., and Sunderland, J. E., 1965, "Heat conduction with freezing or melting," *Appl. Mech. Rev.*, Vol. 18, pp. 951-952.
9. Goodman, T. R., 1964, "Application of integral methods to transient nonlinear heat transfer," *Adv. Heat Transfer*, Vol. 1 pp. 71-79.
10. Evans, G. W., Isaacson, E., and MacDonald, J. K. L., 1950, "Stefan-like problems," *Q. Appl. Math.*, Vol. 8, pp. 312-319.
11. Tao, L. N., 1979, "Free boundary problems with radiation boundary condition," *Q. Appl. Math.*, Vol. 37, pp. 1-10.

12. Tao, L. N., 1979, "On free boundary problems with arbitrary initial and flux conditions," J. Appl. Math. Phys. (ZAMP), Vol. 30, pp. 416-426.
13. Tao, L. N., 1981, "The exact solutions of some Stefan problems with prescribed heat flux," J. Heat Transfer, Vol. 48, pp. 732-736.
14. Cho, S. H., and Sunderland, J. E., 1981, "Approximate temperature distribution for phase change of a semi-infinite body," J. Heat Transfer, Vol. 103, pp. 401-403.
15. Lightfoot, N. M. H., 1929, "Solidification of molten steel," Proc. Lond. Math. Soc., Ser. 2, Vol. 31, pp. 97-116.
16. Kenneth, A. R., and Latif, M. J., 1971, "Heat conduction with melting or freezing in a corner," J. Heat Transfer, Vol. 91, pp. 101-109.
17. Kolodner, I. I., 1956, "Free boundary problem for the heat equation with applications to problems of change of phase," Comm. Pure Appl. Maths., Vol. 9, pp. 1-31.
18. Elliot, C. M., and Ockendon, J. R., 1982, Weak and Variational Methods for Moving Boundary Problems, Research Notes in Mathematics 59, Pitman Advanced Publishing Program, Boston.
19. Ockendon, J. R., 1974, "Technique of analysis," Moving Boundary Problems in Heat Flow and Diffusion, Proc. Conf. Univ. Oxford, March 25-27, pp. 103-110.
20. Boley, B. A., 1974, "The embedding technique in melting and solidification problems," Moving Boundary Problems in Heat Flow and Diffusion, Proc. Conf. Univ. Oxford, March 25-27, pp. 150-172.
21. Sikarskie, D. L., and Boley, B. A., 1965, "The solution of a class of two-dimensional melting and solidification problems," Int. J. Solids Struct., Vol. 1, pp. 203-234.
22. Duda, J. L., and Vrentas, J. S., 1969, "Perturbation solutions of diffusions-controlled moving boundary problems," Chem. Eng. Sci., Vol. 24, pp. 461-470.
23. Pedroso, R. I., and Domoto, G. A., 1973, "Inward spherical solidification-solution by the method of strained coordinates," Int. J. Heat Mass Transfer, Vol. 16, pp. 1037-1043.
24. Pedroso, R. I., and Domoto, G. A., 1973, "Exact solution by perturbation method for planar solidification of a saturated liquid with convection at the wall," Int. J. Heat Mass Transfer, Vol. 16, pp. 1816-1819.

25. Riley, D. S., Smith, F. T., and Poots, G., 1974, "The inward solidification of spheres and circular cylinders," *Int. J. Heat Mass Transfer*, Vol. 17, pp. 1507-1516.
26. Weinbaum, S., and Jiji, L. M., 1977, "Singular perturbation theory for melting or freezing in finite domains not at the fusion temperature," *J. Appl. Mech.*, Vol. 44, pp. 25-30.
27. Jiji, L. M., and Weinbaum, S., 1978, "Perturbation solutions for melting or freezing in annular regions initially not at the fusion temperature," *Int. J. Heat Mass Transfer*, Vol. 21, pp. 581-592.
28. Charach, Ch., and Zoglin, P., 1985, "Solidification in a finite, initially overheated slab," *Int. J. Heat Mass Transfer*, Vol. 28, pp. 2261-2268.
29. Tokuda, N., 1986, "An asymptotic, large time solution of the convection Stefan problem with surface radiation," *Int. J. Heat Mass Transfer*, Vol. 29, pp. 135-143.
30. Yao, L. s., and prusa, J., 1989, "Melting and freezing," *Adv. Heat Transfer*. Vol. 19, pp. 1-95.
31. Yang, K. T., 1967, "Formation of ice in plane stagnation flow," *Appl. Sci. Res.*, Vol. 17, pp. 377-380.
32. Duda, J. L., Malone, M. F., Notter, R. H., and Vrentas, J. S., 1975, "Analysis of two-dimensional diffusion-controlled moving boundary problems," *Int. J. Heat Mass Transfer*, Vol. 18, pp. 901-910.
33. Saitoh, T., 1978, "Numerical method for multidimensional freezing problems in arbitrary domains," *J. Heat Transfer*, Vol. 100, pp. 294-299.
34. Sparrow, E. M., Ramadhyani, S., and Patankar, S. V., 1978, "Effect of subcooling on cylindrical melting," *J. Heat Transfer*, Vol. 100, pp. 395-402.
35. Tein, L. C., and Churchill, S. W., 1965, "Freezing front motion and heat transfer outside an infinite, isothermal cylinder," *AIChE J.*, Vol. 11, pp. 790-793.
36. Hill, J. M., and Kucera, A., 1983, "Freezing a saturated liquid inside a sphere," *Int. J. Heat Mass Transfer*, Vol. 26, pp. 1631-1637.
37. Murray, W. D., and Landis, F., 1959, "Numerical and machine solutions of transient heat-conduction problems involving melting or freezing; Part I-Method of analysis and sample solutions," *J. Heat Transfer*, Vol. 81, pp. 106-112.

38. Landau, H. G., 1950, "Heat conduction in a melting solid," *Q. Appl. Math.*, Vol. 8, pp. 81-94.
39. Rogerson, J. E. Chayt, G. A., 1971, "Total melting time in ablating slab problem," *J. Appl. Phys.*, Vol. 42, No. 7, pp. 2711-2713.
40. Vallerani, E., 1974, "Integral technique solution to a class of simple ablation problems," *Meccanica*, Vol. 9, pp. 94-101.
41. Zien, T. F., 1978, "Integral solutions of ablation problems with time-dependent heat flux," *AIAA J.*, Vol. 16, No. 12, pp.
42. Biot, M. A., Agrawal, H. C., 1964, "Variational analysis of ablation for variable properties," *J. Heat Transfer*, pp. 437-442.
43. Prasad, A., 1980, "Radiative melting of materials with melt removal," *J. Spacecraft and Rockets*, Vol. 17, No. 5, pp. 474-477.
44. Prasad, A., 1979, "Melting of solid bodies due to convective heating with the removal of melt," *J. Spacecraft and Rockets*, Vol. 16, No. 6, pp. 445-448
45. Blackwell, B. F., 1988, "Numerical prediction of one-dimensional ablation using a finite control volume procedure with exponential differencing," *Numer. Heat Transfer*, Vol. 14, pp. 17-34.
46. Hogge, M., Gerrekens, P., 1985, "Two-dimensional deforming finite element methods of surface ablation," *AIAA J.*, Vol. 23, No. 3, pp. 465-472.
47. Masters, J. I., 1956, "Problem of intense surface heating of a slab accompanied by change of phase," *J. Appl. Phys.*, Vol 27, pp. 477-484.
48. Abakian, H., Modest, M. F., 1988, "Evaporative cutting of a semitransparent body with a moving CW laser," *J. Heat Transfer*, Vol. 110, pp. 924-930.
49. Modest, M. F., Abakian, H., 1986, "Evaporative cutting of a semi-infinite body with a moving CW laser," *J. Heat Transfer*, Vol. 108, pp. 602-607.
50. Dabby, F. W., 1972, "High-intensity laser-induced vaporization and explosion of solid material," Vol. QE-8, No. 2, pp. 106-111.
51. Stolz, G. J., 1960, "Numerical solutions to an inverse problem of heat conduction for simple shape", *J. Heat Transfer*, Vol. 82, pp. 20-26.

52. Beck, J. V., 1970, "Surface heat flux determination using an integral method," Nucl. Eng. Des., Vol. 1, pp. 170-178.
53. Burggraf, D. R., 1964, "An exact solution of the inverse problem in heat conduction theory and application," J. Heat Transfer, Vol. 86, pp. 373-382.
54. Beck, J. V., 1970, "Nonlinear estimation applied to the nonlinear inverse heat conduction problem," Int. J. Heat Mass Transfer, Vol. 13, pp. 703-716.
55. Bass, B. R., Drake, J. B., and Ott, L. J., 1980, "ORMDIN: A finite element program for two-dimensional nonlinear inverse heat conduction analysis," Oak Ridge National Laboratory, NUREG/CR-1709.
56. Hsieh, C. K., and Lin, J., 1986, "Solution of inverse heat-conduction problems with unknown initial conditions," Proc. Eighth Int. Heat Transfer Conf., Vol. 2, pp. 609-614.
57. Nikitenko, N. I., and Kolodnyi, Y. M., 1977, "Numerical solution of the inverse heat-conduction problem for determining thermal constants," Inzh.-Fiz. Zh., Vol. 33, pp. 1058-1061.
58. Beck, J. V., Blackwell, B., and St. Clair, C. R. Jr., 1985, "Inverse Heat Conduction, Ill-posed Problems," Wiley-Interscience, New York
59. Hsieh, C. K., Choi, C. Y., and Liu, K. M., 1989, "A domain extension method for quantitative detection of cavities by infrared scanning," J. Nondestr. Eval., Vol. 8(3), pp. 195-211.
60. Chen, M. M., Pederson, C. O., and Chato, J. C., 1978, "On the feasibility of obtaining three-dimensional information from thermographic measurements," J. Biomech. Eng., Vol. 99, pp. 58-64.
61. Macqueene, J. W., Akau, R. L., Kratz, G. W., Schoenals, R. J., 1981, "Numerical methods and measurements related to welding processes," presented at the 2nd International Conf. on Numerical Methods in Thermal Problems, Venice, Italy.
62. Katz, M. A., and Rubinsky, B., 1984, "An inverse finite element technique to determine the change of phase interface location in one-dimensional melting problems," Numer. Heat Transfer, Vol. 7, pp. 269-283.
63. Landram, C. S., 1983, "Measurement of fusion boundary energy transport during arc welding," J. Heat transfer, Vol. 105, pp. 550-554.
64. Rubinsky B., Shitzer A., "Analytic solutions to the heat equation involving a moving boundary with applications to the change of phase problem (the inverse Stefan problem)," J. Heat

Transfer, Vol. 100, pp. 300-304.

65. Hsieh, C. K., and Choi, C. Y., 1992, "A general analysis of phase change storage for solar energy application," J. Solar Energy Eng., J. Solar Energy Eng., Vol. 114, pp. 203-211.
66. Hsieh, C. K., Choi, C. Y., and Kassab, A. J., 1992, "Solution of Stefan problems by a boundary element method," Boundary Element Technology VII, C. A., Brebbia and M. S. Ingber, eds., Computational Mechanics Publications, Southampton, UK, pp. 473-490.
67. Hsieh, C. K., "Unification of source-and-sink method and boundary element method for the solution of potential problems," J. Heat Transfer (in press).
68. Morse, P. M., and Feshbach, H., 1953, Methods of Theoretical Physics, McGraw-Hill, New York.
69. Ozisik, N. M., 1980, Heat Conduction, Wiley, New York.
70. Hsieh, C. K., and Shang, H., 1988, "Solution of boundary value heat conduction problems with variable convective coefficients by a boundary condition dissection method, Nucl. Eng. Design, Vol. 110, pp. 17-31.
71. Brebbia, C. A., 1980, The Boundary Element Techniques in Engineering, Newnes-Butterworth, Boston.
72. Patel, P. D., 1968, "Interface conditions in heat conduction problems with change of phase," AIAA J., Vol. 6, pp. 2454.

BIOGRAPHICAL SKETCH

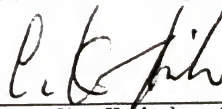
Mehdi Akbari, was born on April 23, 1954, in Hamedan, Iran. He is the first son of six children in the Akbari family.

Mehdi attended Hamedan public schools and was graduated from Dr. Shariati High School in May 1973. He entered the Sharif University of Technology, Tehran, Iran, in September 1973 where he received a degree of Bachelor of Science in mechanical engineering in August 1979.

In May 1986, after six years working in the Technical Institutions and National Iranian Oil company in Iran, he entered the Graduate School at the University of Florida and earned an Master of Science with a thesis on the conversion of solar energy to electricity. He was admitted to the candidacy for Ph.D in 1991.

He is married to Vahideh Lamian and has two daughters with the names of Sara and Mona.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



Chung K. Hsieh, Chairman
Professor of Mechanical
Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.




Reza Abbaschian
Professor of Materials Science
and Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



Gerard G. Emch
Professor of Mathematics

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



Roger A. Gater
Associate Professor of
Mechanical Engineering

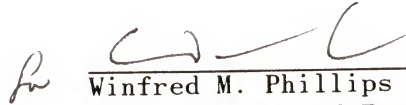
I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



Herbert A. Ingley
Associate Professor of
Mechanical Engineering

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

May 1993



Winfred M. Phillips
Dean, College of Engineering

Madelyn M. Lockhart
Dean, Graduate School